2(m1α)

NASA TECHNICAL MEMORANDUM

GENERALIZED AERODYNAMIC COEFFICIENT TABLE STORAGE, CHECK-OUT AND INTERPOLATION FOR AIRCRAFT SIMULATION

Frank Neuman and Neil Warner

Ames Research Center
Moffett Field, California 94035

(NASA-TM-X-62229) GENERALIZE AERODYNAMIC COEFFICIENT TABLE STORAGE, CHECKOUT AND INTERPOLATION FOR AIRCRAFT SIMULATION (NASA) 107 p HC $7.50 CSCL 01A

N73-16983

Unclas
G3/01 62296

January 1973

108

CONTENTS

GENERALIZED AERODYNAMIC COEFFICIENT TABLE STORAGE, CHECK-OUT AND INTERPOLATION FOR AIRCRAFT SIMUALTION

SUMMARY

For the last three years the set of programs described in this paper has been used for rapidly introducing, checking out and very efficiently using aerodynamic tables in complex aircraft simulations.

The preprocessor program reads in tables with different names and dimensions and stores them on disc storage according to the specified dimensions. The tables are read in from IBM cards in a format which is convenient to the engineer or person who reduces the data from the original graphs. During table processing, new auxiliary tables are generated which are required for table cataloging and for efficient interpolation. In addition, DIMENSION statements for the tables as well as READ statements are punched so that they may be used in other programs for readout of the data from disc without chance of programming errors. Besides punched card output, there is also a line printer output which consists of the properly labelled tables. For quick data checking graphical output for all tables is provided in a separate program.

Linear interpolation routines were written which required only one search per argument, independent of the number of lists for each argument. The linear interpolation routines have been written in FORTRAN to interpolate 1 dimensional, 2 dimensional, and 3 dimensional tables.

The programs described so far have been written to be used for aircraft simulation on the IBM 360. For piloted simulations of the EAI 8400 computer the same set of data is required. However, the data input format is quite differnt. To save time and reduce operator errors a program was written that converts the format for the IBM 360 to a punched card deck in the EAI 8400 compatible format.

INTRODUCTION

The programs that will be described in the following sections are designed to speed up the data storage, data check-out, and table look-up for aerodynamic coefficient tables used in aircraft automatic control simulations on the IBM 360 and for piloted simulations on the EAI 8400.

On the 360 everytime a new aircraft or spacecraft had to be simulated new data processing subroutines had to be written, while the 8400 interpolation is not too efficient in its argument searches. The new table preprocessor program that will be described does not require any re-writing as function of the table names, table lengths, or table dimensions. The new interpolation program performs only one search per argument independent of the number of argument lists required for the set of tables. This is further explained below.

For an elaborate 6 degree of freedom aircraft simulation, up to 100 tables are required. These tables often have common arguments, for instance 'alpha' the angle of attack, but the list of argument values at which the function values are stored may be different depending on the shape of the functions which are approximated as linear segments. These lists are called break-point lists. During a simulation run each break-point list must be searched for the subscript of the break-point just below the actual argument value. Therefore, many break-point lists may have to be searched even though they concern the same argument. In our table look-up method this disadvantage is over-come by generating master lists of all break-points and searching the master lists only. The subscript of the appropriate break-point for each argument of each table is found in an auxiliary cross reference list. Hence, there is only one search per argument. An inefficient search could also be avoided by expanding tables

by means of interpolation so that they all have the same break-point lists. If one assumes infinite storage, this is the most efficient method of table look-up. However, table expansion may increase the required storage by a factor of 4 to 8 so that a large amount of virtual memory and core swapping my be required on the IBM 360 which would make the program run inefficiently although the calculations would be reduced. In the programs that will be described, the tables are kept at their original size at a cost of a small amount of computation.

At present, data packages for aerodynamic tables for aircraft simulation are prepared separately for the IBM 360 and the EAI 8400. This requires two sets of data decks, each containing different types of data transcription and key punch operator errors;and therefore, it requires two check-out periods. Such check-out is very costly in computer and engineer's time. Therefore a program was written that circumvents double data check-out by converting the data stored and checked out on the on the 360 to a deck of cards in the format that is required for the 8400 for aircraft simulation.

CONSIDERATIONS FOR THE PREPARATION OF TABLES

Often aeronautical tables are generated from graphical data. The following remarks are intended to aid in speeding up the data reduction process and aid in obtaining an efficient table lookup for running simulation programs.

The number of breakpoint lists should be kept small. This can be done by examining all tables with identical parameters and by selecting a set of parameter breakpoints that would permit faithful description of all curves with that set of breakpoints. Then select from this master list a reasonable number of subsets to describe individual curves with a minimum number of breakpoints. The maximum number of breakpoint lists per argument is 20, but considerable storage and running time is saved for the simulation program if this number is kept smaller. Greatest care in reducing the number of breakpoint lists should be taken for the argument that appears most often in the tables, usually the angle of attack.

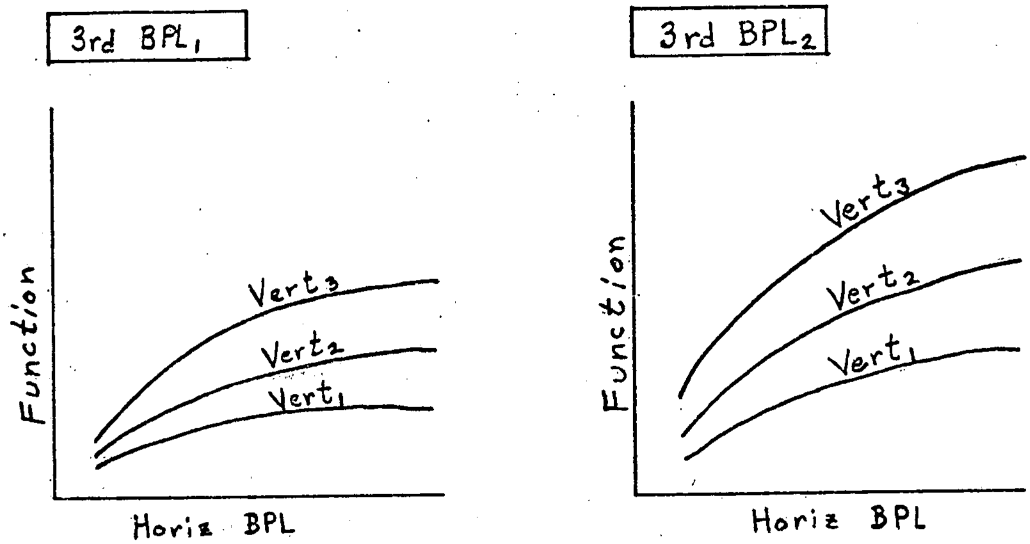
All entries of the tables must be filled. It often happens in graphical data that all curves on a graph do not span the full range of the parameter on the abscissa. It is thought that the engineer can estimate a better extrapolation for the purpose of filling the table than an automatic extrapolation program could. If no prior knowledge is available, or if the simualtion would be invalid if it entered the region of the graph for which no data exist, the engineer can insert extremely large function values, which would make the simulation fail in a very obvious way.

Effort is saved when the tables are constructed from the original graphs by using the same convention as the automatic plotting program. Unless specified differently, for an individual table, the plotting program plots the horizontal breakpoint list as abscissa, the vertical breakpoint list as curve parameter,

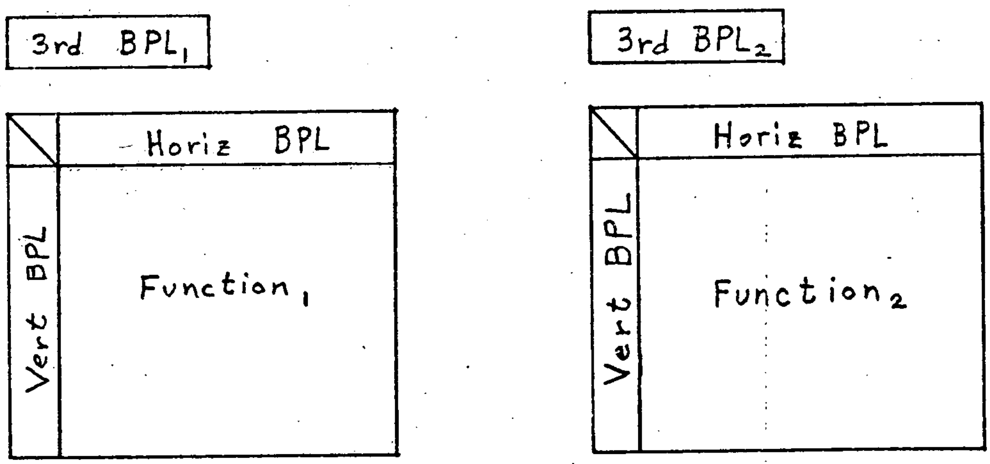
and for a three dimensional table, it uses the third breakpoint list as fixed parameter for each individual graph, see figure 1.

In naming the table it is useful to adhere to a simple convention similar for all tables. A suggested way is to add a T to the name of a function for the tabulated values. Then, e.g. the table name for the lift coefficient CL would be CLT.

The table parameters may be named identical to the names given in the simulation program. The parameter names in the table processor are used only as alphameric data. They aid in labelling the printouts and graphs and are used in error messages in the simulation program.



(a) Graphical data.



(b) Tabulated data.

Figure 1. Relationship between breakpoint lists and standard plotting convention.

TABLE PRE-PROCESSOR

READING IN THE TABLE DATA AND DESCRIPTION OF THE PROGRAM FLOW

The table pre-processor consists of a main program and 5 subroutines all written in FORTRAN (for listings see appendix I). Their operation will be explained by means of a small example. Figure 2 shows the data deck for the example, which includes short 1, 2, and 3 dimensional tables in arbitrary order. The cards are numbered and anotated for the purpose of discussion only.

The main program T READ, reads in the NAMELIST "Input", which contains only the variable IPUNCH. Table storage and card punching is controlled by IPUNCH; when it is '0' only line printer output occurs. It is advisable to run the program in this manner until one is reasonably sure of the correctness of the data format and table dimensions. Finally, set IPUNCH to '1' to store the data on disc and obtain a deck of DIMENSION and READ cards for all tables and auxiliary arrays.

TREAD then calls subroutine TABCOM. TABCOM reads two key cards, (1) and (2), which contain the number of arguments for all tables to be read and the alpha-numerical names of all arguments. For reading each table TABCOM calls TABIN. Each table has a header card, (3), (7), (20). The header card contains, in order, a user's comment field, the table name, the names of the arguments, and the lengths of the individual breakpoint lists which are also the table dimensions. As can be seen, for the examples of the 2 and 3 dimensional tables, the tables are constructed in a conventional fashion with one breakpoint list written horizontally across the top, (9), (15), (21), and the other breakpoint list vertically on the left (first column of numbers). The breakpoint for the third dimension are given at the head of each 2-dimensional subtable, (8), (14). This arrangement requires repetition of breakpoint lists

Card									Explanation
(1)	3								no. of arguments
(2)	ALPHA	DA	MACH						argument names

Card								Explanation
(3)	CM	ALPHA			1	3	1	Table 1 1 dimensional

Card	α	C_m
(4)	0.	-.2
(5)	5.	0.6
(6)	10.	.8

Card								Explanation
(7)	CLDA	ALPHA	DA	MACH	3	4	2	Table 2 3 dimensional

(8) $M_1 = .5$

Card	δ_a \ α	0.	5.	10.
(9)-(10)	-10.	.1	.1	.1
(11)	0.	.3	.2	.1
(12)	15.	.501	.302	.4
(13)	20.	.8	.4	.598

(14) $M_2 = .9$

Card	δ_a \ α	0.	5.	10.
(15)-(16)	-10.	.05	.048	.05
(17)	.0	.146	.1	.098
(18)	15.	.46	.15	.2
(19)	20.	.39	.20	.31

Card							Explanation
(20)	CLMN	DA	ALPHA	3	3	1	Table 3 2 dimensional

Card	α \ δ_a	-9.	10.	18.
(21)-(22)	0.	0.00012	.2001	.41
(23)	6.	.2	.6	.8
(24)	10.	.4	1.0	1.2

Figure 2. Card images with explanations of a sample set of input data to the pre-processor. (Blank cards were inserted for printout clarity only.)

for 3-dimensional tables, but it aids the visual check of the data cards before processing. The corresponding function values are located at the intersection of one element in the vertical breakpoint list and another element in the horizontal breakpoint list.

In addition to the above, TABCOM stores the function elements and saves the name of the table and the names of the arguments associated with the breakpoint sublists. Next, once per table, TABCOM calls TABOUT which is a subroutine that writes out the table. The output for the example in Figure 2 is shown in Figure 3. TABCOM also compares the new breakpoint lists with those already stored and adds them to the already stored breakpoint lists, unless an identical breakpoint is already in storage. Furthermore, TABCOM assembles breakpoint identification lists which identify the required breakpoint lists as function of the table number and first, second, and third table dimensions as appropriate. After all tables have been read in, TABCOM prints out all breakpoint lists as shown in Figure 4(a). Control is then turned back to the main program.

The last job to be performed by the table pre-processor is to generate the lists that are required for a more efficient table look-up. First the main program calls subroutine MLIST. MLIST generates a master list for each argument that occurs in the tables. (A master list is an ordered list of all values that occur in the sublists for one particular argument). These master lists are required for searching each argument that occurs in the tables. The master lists are printed out as shown in Figure 3(b). Finally, subroutine NBLIST is called. This subroutine generates cross reference lists which will allow table look-up by only searching the master list for each argument. After these lists are constructed they are printed out as shown in Figure 5(a) for the example. The NBPL lists define the corresponding element in the breakpoint list for each argument and each table. When the breakpoint list for a given

NUMBER OF VARIABLES 3
ALPHA DA MACH

TABLE 1	CM	(ALPHA)

ALPHA	CM
0.000000	-0.200000
5.000000	0.600000
10.000000	0.800000

TABLE 2	CLDA	(ALPHA , DA , MACH)

MACH = 0.500000

DA \ ALPHA	0.000000	5.000000	10.000000
-10.000000	0.100000	0.100000	0.100000
0.000000	0.300000	0.200000	0.100000
15.000000	0.501000	0.302000	0.400000
20.000000	0.800000	0.400000	0.598000

(a)

Figure 3. Printed tables from the pre-processor.

MACH = 0.900000

DA	ALPHA 0.000000	5.000000	10.000000
-10.000000	0.050000	0.048000	0.050000
0.000000	0.146000	0.100000	0.098000
15.000000	0.460000	0.150000	0.200000
20.000000	0.390000	0.200000	0.310000

TABLE 3 CLMN (DA , ALPHA)

ALPHA	DA -9.000000	10.000000	18.000000
0.000000	0.000120	0.200100	0.410000
6.000000	0.200000	0.600000	0.800000
10.000000	0.400000	1.000000	1.200000

(b)
FIGURE 3 (concluded).

		ALPHA		
1 CM	0.00000	5.00000	10.00000	
3 CLMN	0.00000	6.00000	10.00000	

		DA		
2 CLDA	-10.00000	0.00000	15.00000	20.00000
3 CLMN	-9.00000	10.00000	18.00000	

		MACH
2 CLDA	0.50000	0.90000

(a) Break-point lists, none repeated, with the table number and name where each list first occurred.

MASTER LISTS

4 ALPHA

0.000	5.000	6.000	10.000			

7 DA

-10.000	-9.000	0.000	10.000	15.000	18.000	20.000

2 MACH

0.500	0.900

(b) Break-point master lists with number of elements and name of argument.

Figure 4. Auxiliary arrays from pre-processor.

NBPL LISTS FOR ALPHA

		CM	CLMN
1	0.000	1	1
2	5.000	2	1
3	6.000	2	2
4	10.000	3	3

NBPL LISTS FOR DA

		CLDA	CLMN
1	-10.000	1	-1
2	-9.000	1	1
3	0.000	2	1
4	10.000	2	2
5	15.000	3	2
6	18.000	3	0
7	20.000	4	0

NBPL LISTS FOR MACH

		CLDA
1	0.500	1
2	0.900	2

(a) NBPL cross reference lists.

CROSS REFERENCE LISTS

J	NTDIM	NJ1	NJ2	NJ3	NK1	NK2	NK3	NL1	NL2	NL3
1	1	1			1			3		
2	3	1	2	3	1	1	1	3	4	2
3	2	2	1		2	2		3	3	
table no.	no. dimen.	argument no. for each dimension			break-point list no. for the given arg., 1st, 2nd, 3rd dimen.			length of the break-point lists		

(b) Cross reference lists indicating the correct breakpoint list for each argument of each table.

16 DIMENSION CARDS WERE PUNCHED.

18 READ CARDS WERE PUNCHED.
NOTE *** ON TSS, THESE CARD IMAGES ARE STORED IN THE DATA SET
ASSIGNED TO FT07F001 BY A DDEF CARD.
SEE THE TABLE PROCESSOR USERS GUIDE.

(c) Message giving number of access card images stored.

Figure 5. Final output from the pre-processor.

table does not cover the whole range of the master list, for points outside the range, a '-1' is inserted when the value is below the range and a '0' is inserted in the NBPL list when the value is above the range. (See the DA NBPL list for table CLMN). As will be shown later, this permits the interpolation subroutines to determine, with a single IF statement, whether an argument is outside the range of the table.

Other cross-reference lists that are required are shown in figure 5(b). At the beginning of the main program, under symbol explanation, the meaning of the important variables and tables are given. Sufficient comment cards are inserted into the programs to make them self-explanatory. Also given in the main program are the ranges of the subscripts, which are determined by the dimension statements presently in the program. The program as it stands can handle 25 different independent variables and a maximum of 20 different breakpoint lists per argument. It is of computational and storage advantage if the numbers of breakpoint lists per argument are kept to a minimum. The maximum length of a master list must not exceed 35 elements. These limits, of course, can be changed by simply changing the dimensions of the arrays.

Since the pre-processor has stored all arrays efficiently, the remaining programs could not be written in complete generality. However, the only change that must be made as function of the type of tables to be processed is to take the dimension cards and read statements which are provided by the table pre-processor (see Figure 6) and insert them at the beginning of the main programs. Note that the READ cards must remain in the correct order.

```
      DIMENSION CM        (  3)                                         D  1
      DIMENSION CLDA      (  3,   4,   2)                               D  2
      DIMENSION CLMN      (  3,   3)                                    D  3
      REAL*8 VARNM(   3)                                                D  4
      DIMENSION BPL(   3,   2,   4)                                     D  5
      DIMENSION NLENTH(   3,   2)                                       D  6
      DIMENSION NTABLE(   3,   2)                                       D  7
      REAL*8 TNME(   3)                                                 D  8
      DIMENSION JTN(   3)                                               D  9
      DIMENSION NTDIM(   3)                                             D 10
      DIMENSION NBPL(   3,   7,   2)                                    D 11
      DIMENSION NMAST (   3)                                            D 12
      DIMENSION AMAST(   7,   3)                                        D 13
      DIMENSION NJ( 3,   3)                                             D 14
      DIMENSION NK( 3,   3)                                             D 15
      DIMENSION NL( 3,   3)                                             D 16
      READ (2)CM                                                           1
      READ (2) CLDA                                                        2
      READ (2) CLMN                                                        3
      READ (2) NUMTBL                                                      4
      READ (2) NOVARS                                                      5
      READ (2) VARNM                                                       6
      READ (2) BPL                                                         7
      READ (2) NLENTH                                                      8
      READ (2) NTABLE                                                      9
      READ (2) TNME                                                       10
      READ (2) JTN                                                        11
      READ (2) NTDIM                                                      12
      READ (2) NBPL                                                       13
      READ (2) NMAST                                                      14
      READ (2) AMAST                                                      15
      READ (2) NJ                                                         16
      READ (2) NK                                                         17
      READ (2) NL                                                         18
```

Figure 6. Listing of the DIMENSION and READ cards provided by the pre-processor. These cards must be inserted into the graphics program, the 8400-card punch program, and the interpolation program.

DATA VERIFICATION

Although the tables are printed out in easily readable form, it is still tedious to compare the data with the original graphical aeronautical data. This process is simplified by graphical output of all tables that have been stored. A main plotting program and some subroutines are included in this table processing package to plot the tables stored by the pre-processor. Hardcopy 8 x 8 - inch plots are produced on the S-C4020 cathode-ray-tube plotter.

Sample plots of the example tables described previously are seen in figures 7(a) through 7(g). Figure 7(a) shows a 1 dimensional table, CM. Figures 7(b) and 7(c) show a 3 dimensional table, CLDA, which is plotted with the variables in order as stored by the pre-processor and has been automatically scaled on the stored data. Figures 7(d), 7(e) and 7(f) show the table CLDA plotted where the plotting program options are used. The plotting order has been changed so that the variable DA is now plotted as the abscissa. The 3 variables can be plotted in any order the user desires.

Another option illustrated is specification of the minimum and maximum values of the function CLDA to be used on the graph. Figure 7(g) shows the format for a 2 dimensional table and further illustrates results of specifying the minimum and maximum graph values for the function CLMN and the variable DA. Instructions for specifying the options desired and the cards used for figure 7 are described in Appendix VI.

Listings of the programs to produce the data verification graphs are presented in Appendix II. Detailed instructions for their use are in Appendix VI.

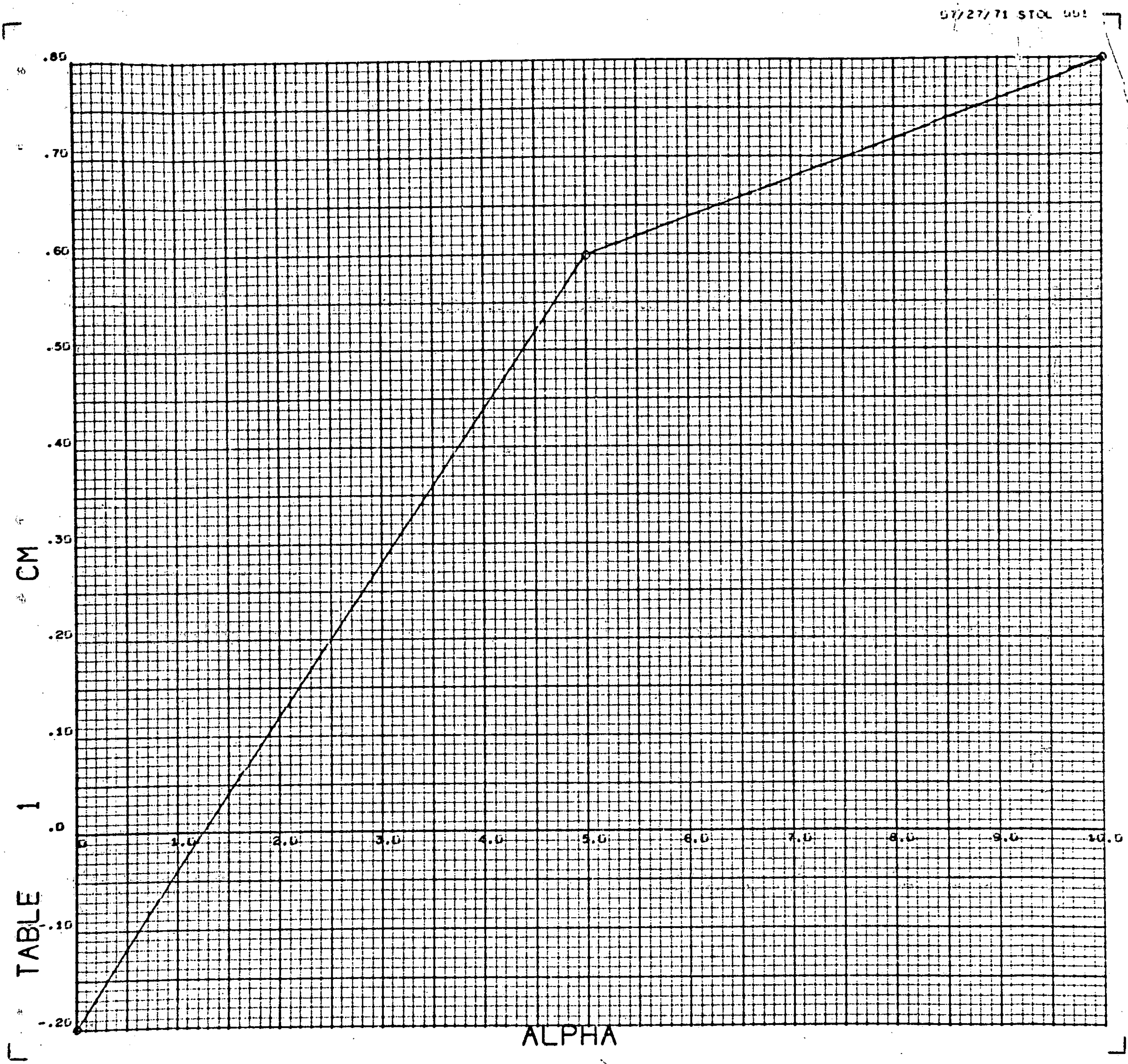


Figure 7(a). One dimensional table plotted as stored.

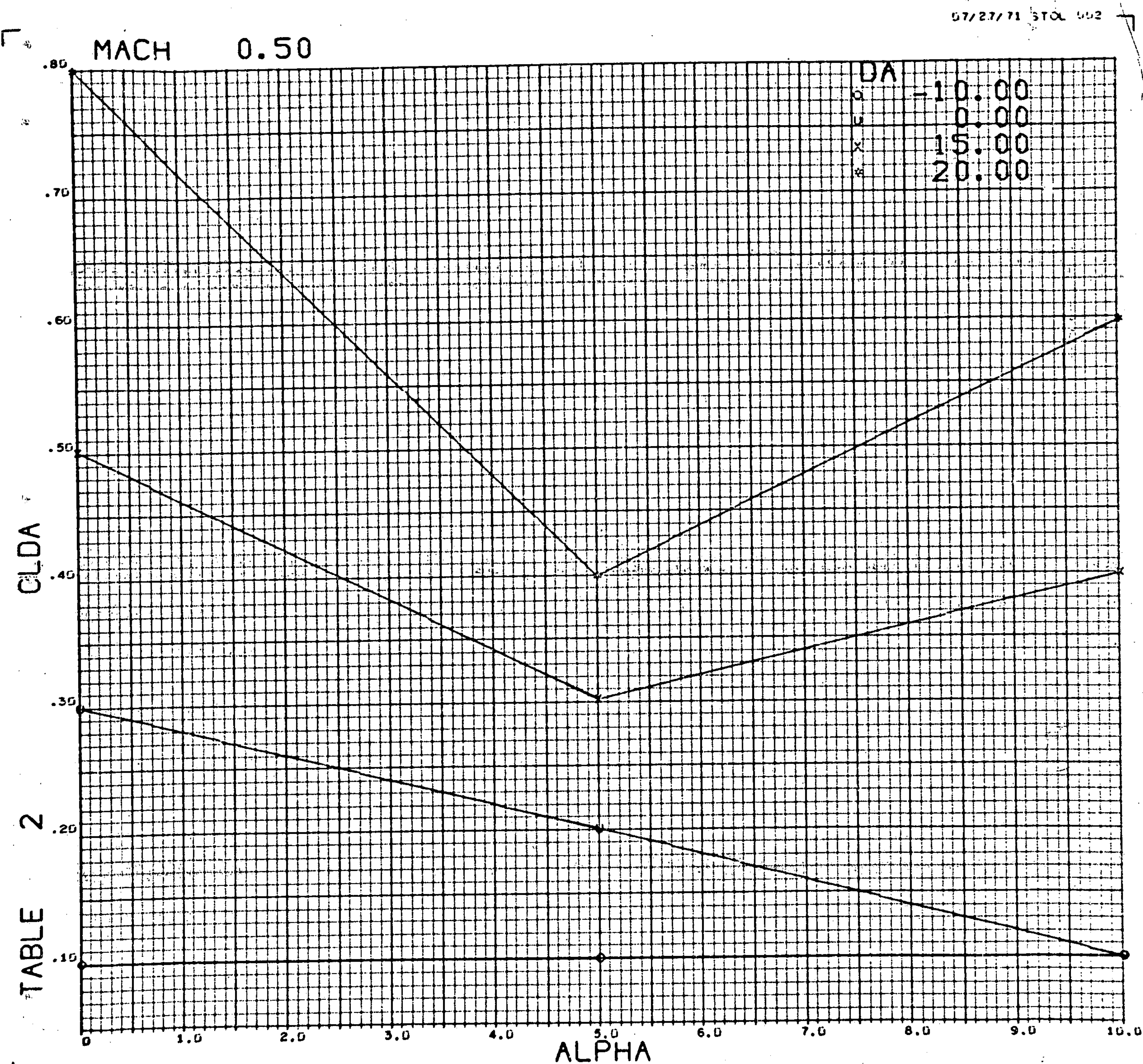


Figure 7(b). Three dimensional table (part 1) plotted as stored, and automatically scaled.

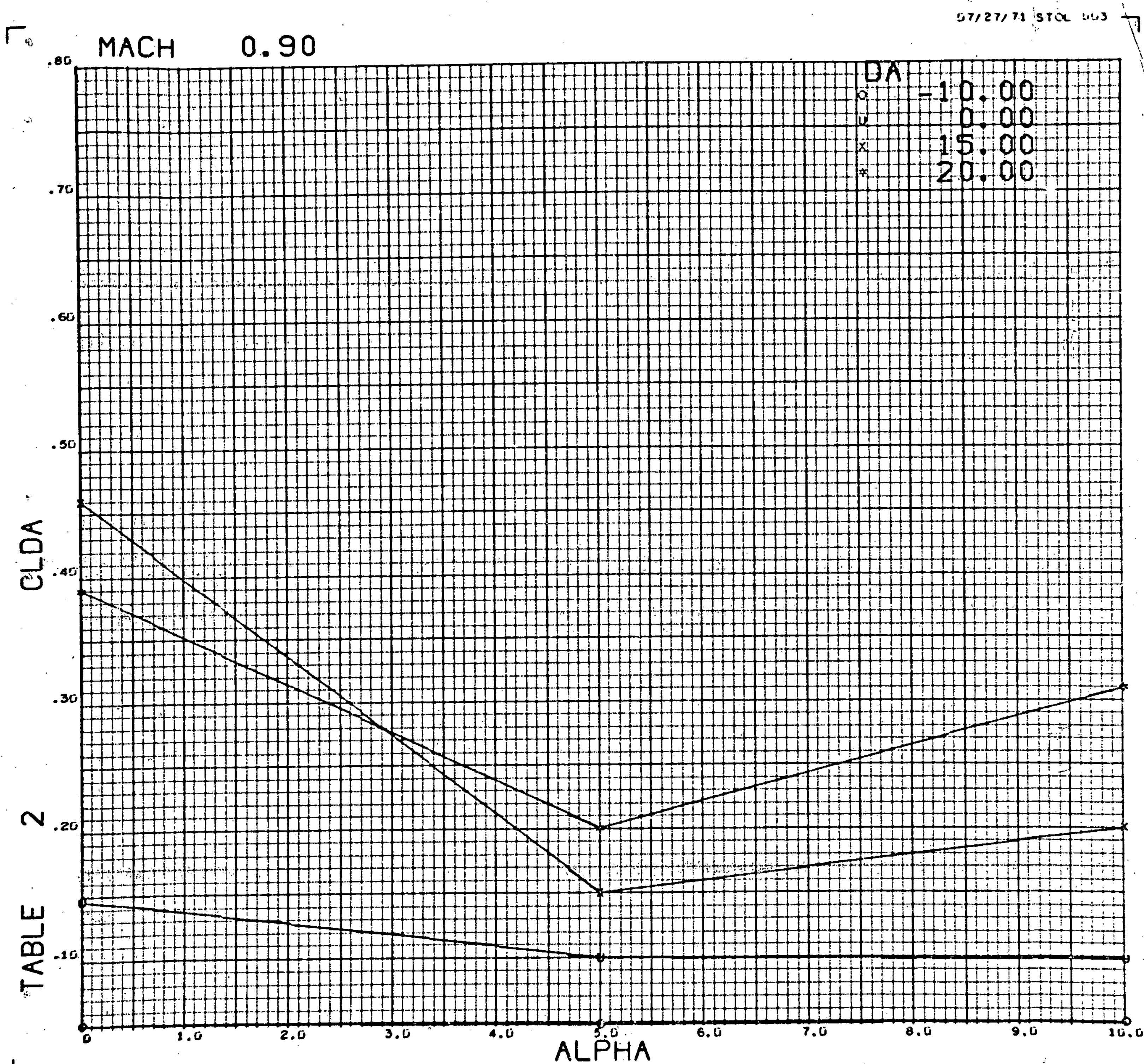


Figure 7(c). Three dimensional table (part 2) plotted as stored and automatically scaled.

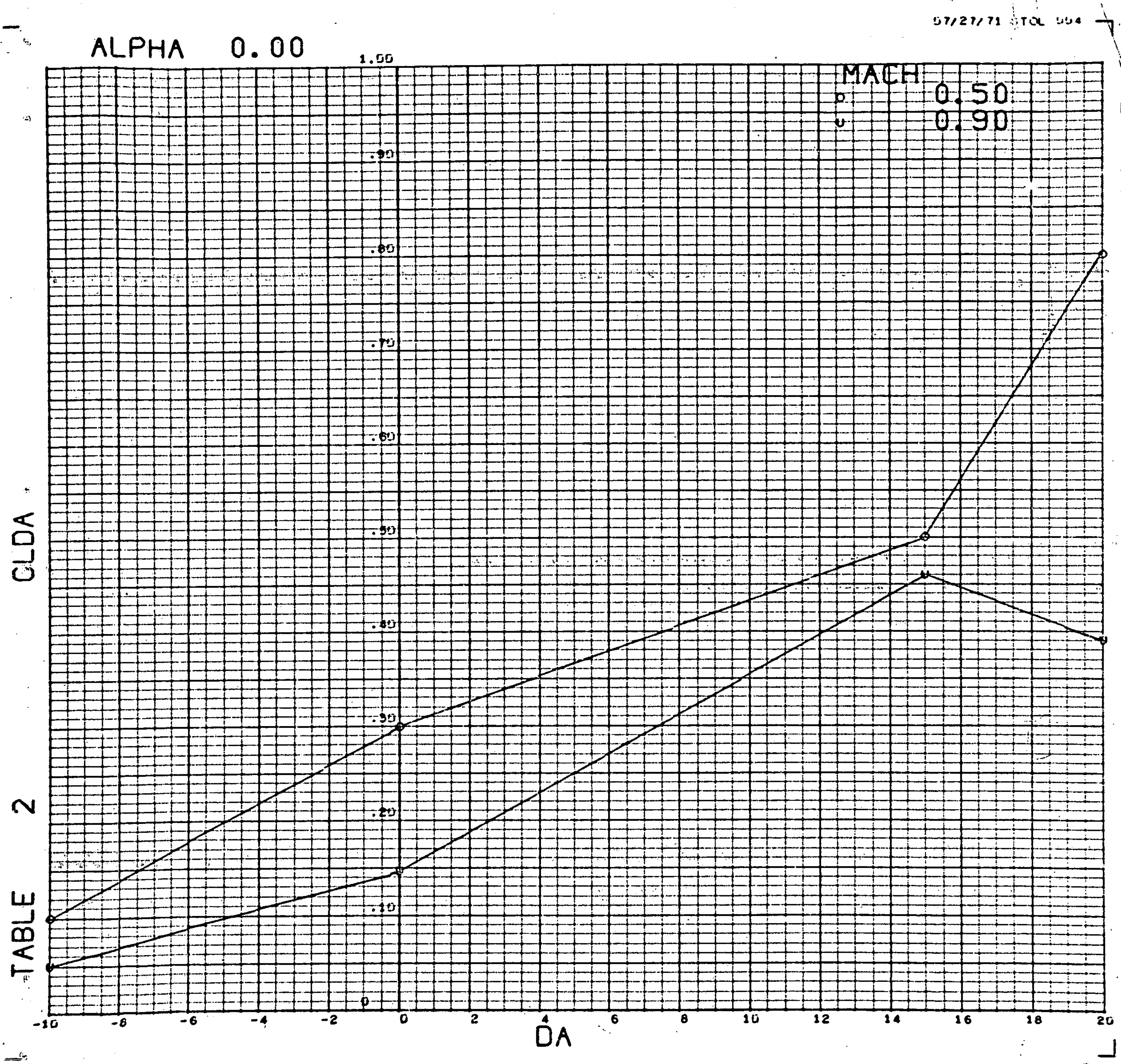


Figure 7(d). Three dimensional table (part 1) showing a change in plotting order from the graphs of figure 5(b) and (c). New minimum and maximum values of the function CLDA were also specified.

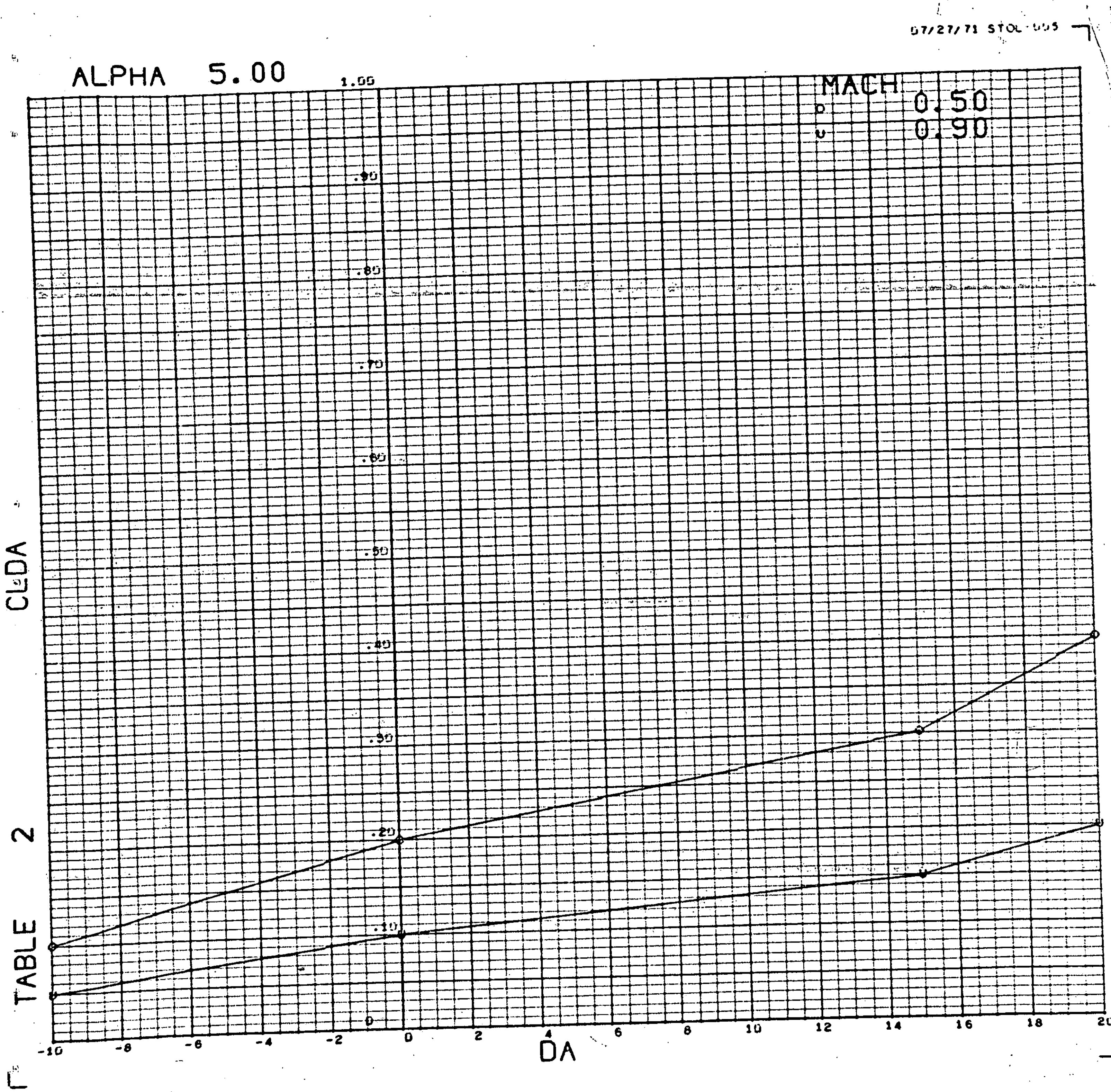


Figure 7(e). Three dimensional table (part 2).

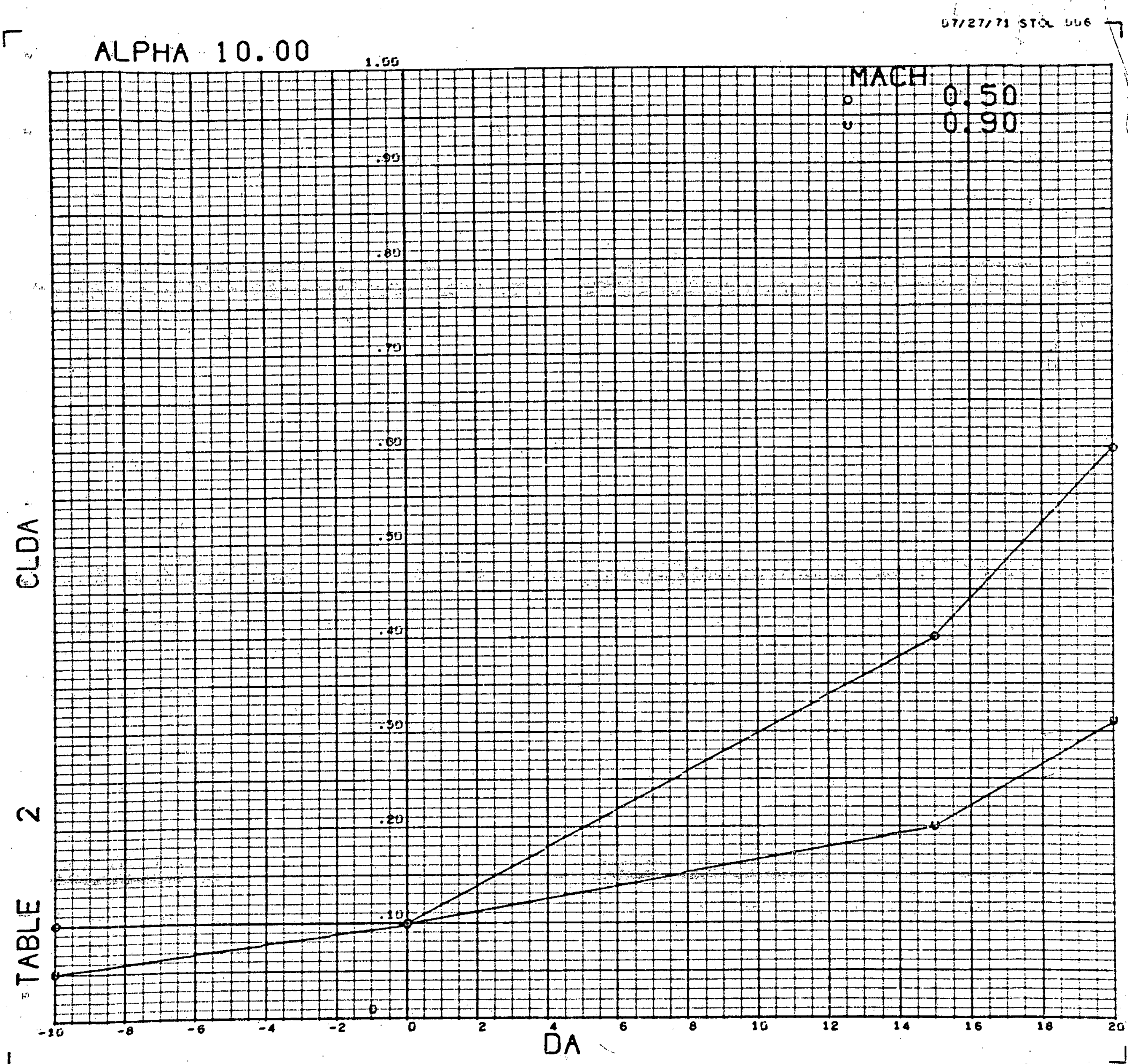


Figure 7(f). Three dimensional table (part 3).

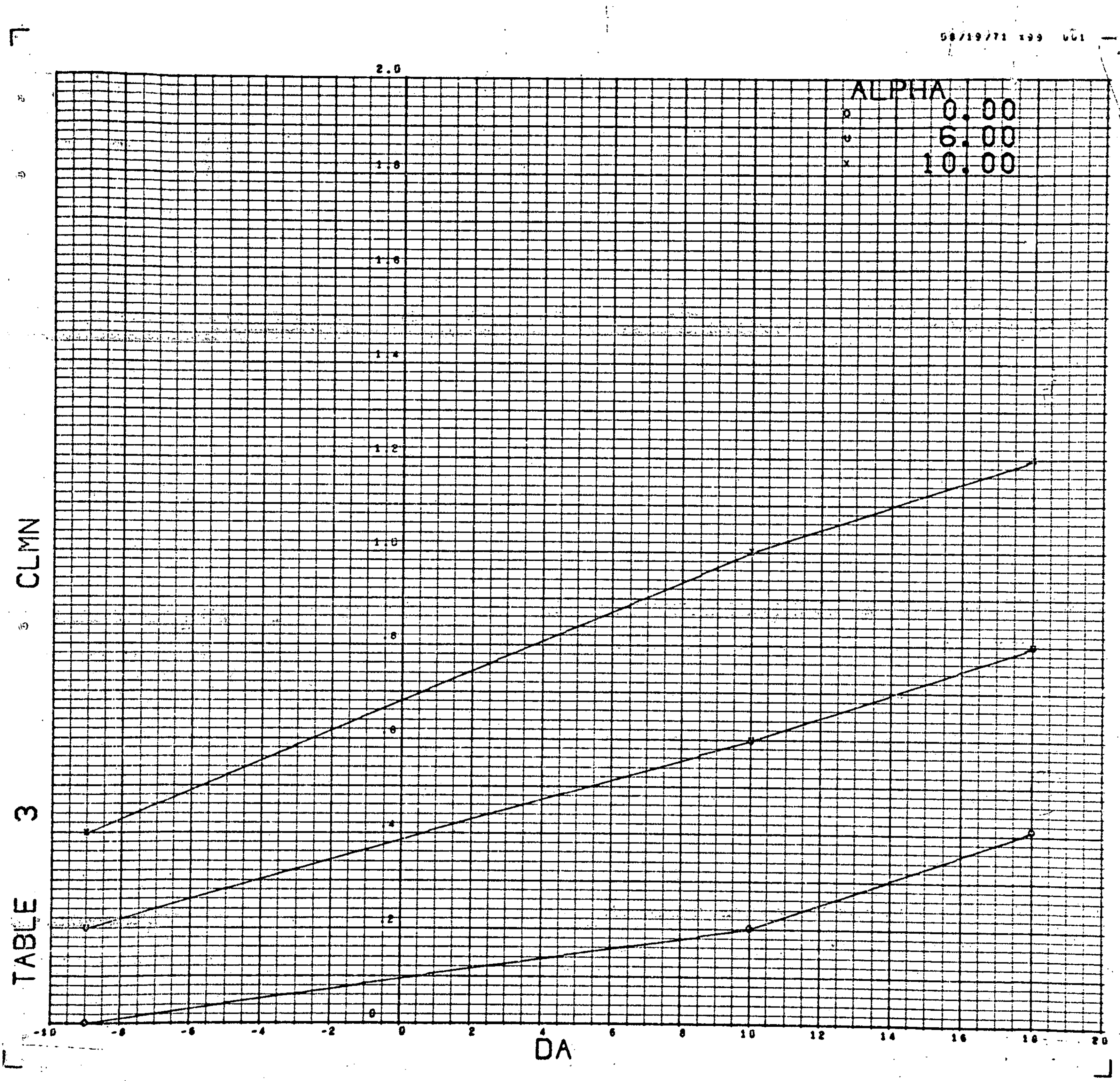


Figure 7(g). Two dimensional table showing results of user specification of minimum and maximum values for both axes.

CHANGE FROM 360 TO 8400 INPUT PUNCH DATA CARDS

As has been discussed in the introduction, a different set of punched table data cards is needed for the 8400. In the 8400 the breakpoint lists must be entered separately and identified with different names for each breakpoint list although they concern the same argument. The function values for each table must be entered separately in a specific format which separates individual numbers by commas. Also, a slash is used as termination indicator for each table. The program that does the conversion from the 360 to the 8400 format consists of a main program (P8400) and a subroutine (FTOB) which converts floating point numbers to a short alpha-numeric array representing the same number in the shortest possible format followed by a comma.

As shown in Figure 8 for the example tables, the output of the program provides a listing and a deck of cards with the following functions. There is one THRUPUT card for each table, which gives the name of the table as well as the arguments in the order in which they are stored. Each THRUPUT card printed image is followed by a line of comment which will aid in associating the names of the breakpoint lists on the THRUPUT cards with the proper breakpoint list. Secondly, all breakpoint lists are given and the name of the argument for each breakpoint list is defined. In addition, a comment card is included. Since each breakpoint list must have a different name in the 8400 program, a suggestion for accomplishing this is given at the head of the printout for this program, as shown in Figure 9.

```
NONE OF THE BREAKPOINTLISTS PUNCHED IS DUPLICATED
AS INITIALLY PUNCHED ALL BREAKPOINTLISTS ARE CALLED BY THE SAME NAME,
 SINCE WE CANNOT PREDICT WHAT THE NAMES OF THE INDIVIDUAL BREAKPONTLISTS WILL BE.
SUGGESTION=CHANGE CARDS AS PER EXAMPLE
FROM=    THRUPUT FAFB(CJA      ,DA      )
TO  =    THRUPUT FAFB(CJA1     ,DA1     )
ACCORDING TO THE NUMBER OF THE BREAKPOINTLIST FOR THE  GIVEN TABLE.
THIS NUMBER IS GIVEN IN THE COMMENT LINE.
THE VARBPT CARDS MUST BE CHANGED IN A SIMILAR MANNER
```

FIGURE 9. Printout from Data Conversion Program

```
      THRUPUT CM        (ALPHA      )
   COMMENT, BK PT LIST NO.    1
      THRUPUT CLDA      (ALPHA     ,DA         ,MACH       )
   COMMENT, BK PT LIST NOS.   1            1            1
      THRUPUT CLMN      (DA        ,ALPHA      )
   COMMENT, BK PT LIST NOS.   2            2
*THIS IS BK PT LST NO.    1 FOR VARIABLE ALPHA
ALPHA    VARBPT
                     .0,5.,10.
                     /
*THIS IS BK PT LST NO.    2 FOR VARIABLE ALPHA
ALPHA    VARBPT
                     .0,6.,10.
                     /
*THIS IS BK PT LST NO.    1 FOR VARIABLE DA
DA       VARBPT
                     -10.,.0,15.,20.
                     /
*THIS IS BK PT LST NO.    2 FOR VARIABLE DA
DA       VARBPT
                     -9.,10.,18.
                     /
*THIS IS BK PT LST NO.    1 FOR VARIABLE MACH
MACH     VARBPT
                     .5,.9
                     /
CM       POINTS
                     -.2,.6,.8
                     /
CLDA     POINTS
*        MACH        = 0.50000
                     .1,.1,.1
                     .3,.2,.1
                     .501,.302,.4
                     .8,.4,.598
*        MACH        = 0.90000
                     .05,.048,.05
                     .146,.1,.098
                     .46,.15,.2
                     .39,.2,.31
                     /
CLMN     POINTS
                     .00012,.2001,.41
                     .2,.6,.8
                     .4,1.,1.2
                     /
```

Figure 8. Line printer output from the 8400 data conversion program.

One appends a number to the name of the variable for each breakpoint list according to the number of the breakpoint list for the given variable. The comments automatically identify which breakpoint list number belongs to the variable for the given table. One must append this number to the name of the variable in the THRUPUT card. We intentionally did not automatize this process, since the choice of the name for the breakpoint list is up to the user, and he may choose completely different names for the individual breakpoint lists.

There are no changes required for card output of the function values themselves. The proper comment cards are inserted for 3 dimensional tables. The function elements are ordered as specified in the THRUPUT statements with the fastest argument on the left and slower varying arguments successively to the right. The punched cards are identical to the line printer output, but in the 026 card punch format which is required for the EAI 8400 computer.

Appendix VII describes use of this program and shows a set of control cards for the TSS System. Again, the main program, shown in Appendix III must be recompiled for each card punch job because of the DIMENSION and READ statements.

With the availability of the data conversion program, the table pre-processor, and the plotting program it is hoped that in the future all function tables will be punched in the manner required for the pre-processor and coverted to the 8400 format by the conversion program. This should provide a faster data check-out, even if were not desired at present to put the aircraft simulation on the IBM 360. Also, hard copy graphical documentation of the tables that are being used for simulation are obtained. If, at a later date, the program for an aircraft simulation is to be transferred to the 360, further benefit is derived from using the simple data card format for the pre-processor.

THE INTERPOLATION SUBROUTINES

The interpolation subroutines are used to interpolate 1, 2 and 3 dimensional tables. Interpolation efficiency is increased by a search subroutine (SEARCH) which takes advantage of the cross-reference list NBPL, see Figure 5(a). At the beginning of each programming cycle the master lists are searched by the subroutine SEARCH. This subroutine determines the subscript of the master list breakpoint if the nearest value is lower than the argument and calculates the proportionality constants PFAC for all breakpoint lists which are needed to interpolate the functions.

$$\text{PFAC} = \frac{\text{Argument} - \begin{bmatrix}\text{Nearest lower} \\ \text{break-point}\end{bmatrix}}{\begin{bmatrix}\text{Nearest higher} \\ \text{break-point}\end{bmatrix} - \begin{bmatrix}\text{Nearest lower} \\ \text{break-point}\end{bmatrix}}$$

First, however, the limits of the master lists are checked against the actual value of each argument. The argument is reset to the nearest limit when it is outside the limits and PFAC is set to zero. In this case a message is printed out indicating the facts.

The acutal interpolation of the tables is accomplished by three function subporgrams. FOUTlD interpolates one dimensional tables; FOUT2D interpolates two dimensional table; and FOUT3D interpolates three dimensional tables. The three subprograms and interpolation equations are structurally similar. The one dimensional interpolation equation is:

$$F = F\begin{pmatrix}\text{at nearest lower} \\ \text{break-point}\end{pmatrix} + \text{PFAC}\left[F\begin{pmatrix}\text{at nearest higher} \\ \text{break-point}\end{pmatrix} - F\begin{pmatrix}\text{at nearest lower} \\ \text{break-point}\end{pmatrix}\right]$$

For the purpose of storage efficiency the subrpograms are not written in complete generality. The dimensions in the DIMENSION statements must be changed to match the array sizes required for a specific aircraft simulation. Since the breakpoint lists of some tables do not cover the complete range of the master lists, each interpolation subprogram must check if the corresponding arguments

are within their limit. This is accomplished very quickly with the aid of the cross reference list NBPL, as explained in appendix VIII. Auguments outside the breakpoint list limit are reset to the appropriate limit for that table.

For increased computational efficiency by a factor of four, the two and three dimensional table interpolation subprograms were written in one dimensional array lookup form. Listings of the search and interpolation subprograms are given in appendix IV. These subprograms must be recompiled for each aircraft simulation to have the proper DIMENSION statements. Source programs may be obtained from the authors. Also, the READ cards from the pre-processor program have to be incorporated into the simulation to read in the tables. In appendix VIII an example is given of the programming required.

CONCLUSIONS AND COMMENTS

A unified approach has been presented to table processing, checkout and table interpolation for aircraft simulation. Complete program listings are included in this report for documentation and to permit changes and additions by the user. The programs were specifically written with Ames computers and computer operation in mind, but portions of the programs could easily be adapted to other users' needs.

Listings of the programs used in this table processing package are given in Appendices I through IV. Detailed information for their use on the TSS system is given in appendices V through VIII. For further information on the control cards, users are referred to "A Guide to TSS/360 Batch Job Processing" and other TSS manuals available from the Computational Division library.

The computer programs comprising this table processing package have been compiled and stored in the authors' job library. This job library has been set up to be shared by all TSS users. Information on accessing it is given in Appendix V.

APPENDIX I

PRE-PROCESSOR PROGRAM LISTINGS

This appendix contains listings of the computer programs used to process and store the aero tables. The programs (main and subroutines) are in the following order:

1. TREAD (main program)
2. TABCOM
3. TABIN
4. TABOUT
5. NBLIST
6. MLIST

```
C TREAD$$
C   MAIN PROGRAM FOR CATALOGING AND STORING AERO TABLES
C
C
C SYMBOL EXPLANATION
C NUMTBL= NUMBER OF TABLES IN TOTAL
C NOTABL= NUMBER OF A SPECIFIC TABLE READ IN
C NOVARS=NUMBER OF VARIABLES ,FUNCTION ARGUMENTS.
C VARNM(J)=APHANUMERIC NAME OF THE J TH VARIABLE
C JTN(J)=NUMBER OF BREAK POINT SUBLISTS FOR THE J TH VARIABLE
C NLENTH(J,K)=NUMBER OF ELEMENTS IN THE K TH SUBLIST FOR THE JTH VAR
C NTABLE(J,K)=TABLE NUMBER TO WHICH THE KTH SUBLIST FOR THE J TH VAR
C              BELONGS
C TNME(NTABLE(J,K))=TABLE NAME CORRESPONDING TO  A GIVEN TABLE NUMBER
C BPL(J,K,L) K TH BREAK POINT LIST FOR J TH VARIABLE ,L ELEMENTS
C  NJ(M,N) = NUMBER OF THE VARIABLE FOR THE M TH DIMENSION OF THE N TH
C    TABLE --- SUBSCRIPT J IN BPL ARRAY
C  NK(M,N) = NUMBER OF THE BREAKPOINT SUBLIST FOR THE M TH DIMENSION OF
C    THE N TH TABLE --- SUBSCRIPT K IN BPL ARRAY
C  NL(M,N) = LENGTH OF THE BREAKPOINT SUBLIST FOR THE M TH DIMENSION OF
C    THE N TH TABLE --- SUBSCRIPT L IN BPL ARRAY
C NMAST(J)=NUMBER OF ELEMENTS IN MASTER LIST OF J TH VARIABLE
C AMAST(NMAST(J),J)=MASTER LIST OF BREAK POINTS FOR THE J TH VARIABLE
C NTDIM(N)=DIMENSION OF THE N TH TABLE
C*******************************************************************
C  RANGE OF SUBSCRIPTS                          MAX VALUES  DECLARED
C J=1,NOVARS                                    25
C K=1,JTN(J)                                    20
C L=1,NLENTH(J,K)                               20
C NMAST(J)=MASTERLIST MAX LENGTH                35
C MAX NUMBER OF TABLES                         200
C*******************************************************************
C
      COMMON VNAME,VARNM,TNAME,TNME,NVLGTH,BPSUBL,FUNC
     1,INEND,NOTABL
     2       ,JTN,NLENTH,NTABLE,BPL,NOVARS
     3,LISTDM,NCARD,NDCARD,IPUNCH,NUMTBL
      COMMON/TABLE/NTDIM,NL,NK,NJ
      DIMENSION NTDIM(200),NL(3,200),NK(3,200),NJ(3,200)
      REAL*8 VARNM,VNAME,TNME,TNAME
      DIMENSION VNAME(3),NVLGTH(3),BPSUBL(25,3),FUNC(20,20,10)
      DIMENSION VARNM(25) ,JTN(25), NLENTH(25,20),NTABLE(25,20)
     1,BPL(25,20,20),TNME(200)
      DIMENSION AMAST(35,25),NMAST(25),TEMP(350)
      DIMENSION TMAST(350)
      DATA MASTMX/0/
```

```
      NAMELIST /INPUT/ IPUNCH
C   START PUNCHED CARD COUNT
      NCARD = 1
      NDCARD = 1
C   READ JOB CONTROL INFO
      READ (5,INPUT)
      DO 60 J=1,10
 60   JTN(J)=0
      CALL TABCOM
      PRINT 41
 41   FORMAT(1H1,12HMASTER LISTS)
C CONSTRUCT MASTER LISTS
      DO 10 J=1,NOVARS
C APPEND SUBLISTS OF THE VARIABLES
      NTL=1
      NT=0
      JT=JTN(J)
      DO 20 K=1,JT
      NT=NT+NLENTH(J,K)
      DO 30 L=NTL,NT
      LMNTL=L-NTL
 30   TEMP(L)=BPL(J,K,LMNTL+1)
 20   NTL=NT+1
C CHECK THAT THE TEMPORARY LIST FITS THE MAXIMUM DECLARED DIMENSION
      IF( NT .GT. 350 ) PRINT 80, NT
  51  FORMAT(1H ,10F8.3)
C GENERATING THE TEMPORARY MASTER LIST
      CALL MLIST(TEMP,NT,TMAST,NMAST(J))
      PRINT 40,NMAST(J),VARNM(J)
 40   FORMAT(1H0,I5,1X,A8)
      NM=NMAST(J)
C PERMANENT STORAGE OF THE MASTER LIST FOR ALL THE ARGUMENTS
      DO 90 K=1,NM
 90   AMAST(K,J)=TMAST(K)
      PRINT 51,(AMAST(K,J),K=1,NM)
 10   CONTINUE
      CALL NBLIST(AMAST,NMAST)
      PRINT 84
 84   FORMAT(1H1,'CROSS REFERENCE LISTS',/,2X,
     1 '  J  NTDIM  NJ1 NJ2  NJ3  NK1  NK2  NK3  NL1  NL2  NL3')
      DO 73 J=1,NUMTBL
C WRITING OUT THE SEARCH LIST
      IF(NTDIM(J)-2) 72,71,70
 72   PRINT 82,J,NTDIM(J),NJ(1,J),NK(1,J),NL(1,J)
 82   FORMAT(1H ,3I5,10X,I5,10X,I5)
      GO TO 73
```

```
 71   PRINT 81,J,NTDIM(J),NJ(1,J),NJ(2,J),NK(1,J),NK(2,J),NL(1,J),NL(2,J
     1)
      GO TO 73
 81   FORMAT(1H ,4I5,5X,2I5,5X,2I5)
 70   PRINT 80,J,NTDIM(J),(NJ(K1,J),K1=1,3),(NK(K2,J),K2=1,3)
     1,(NL(K3,J),K3=1,3)
 80   FORMAT(1H ,12I5)
 73   CONTINUE
C  IF IPUNCH=0, DO NOT PUNCH CARDS
      IF (IPUNCH) 150,150,700
 700  CONTINUE
C       FIND MAX SIZE OFAMAST INDEX
      DO 800 J=1,NOVARS
      IF (NMAST(J) .GT. MASTMX) MASTMX = NMAST(J)
 800  CONTINUE
C       STORE NMAST
      WRITE (2) (NMAST(J),J=1,NOVARS)
      WRITE (7,128) NOVARS,NDCARD
 128  FORMAT (6X,'DIMENSION NMAST (',I3,')',T76,'D',I3)
      NDCARD = NDCARD + 1
      WRITE (7,130) NCARD
 130  FORMAT (6X,'READ (2) NMAST',T77,I3)
      NCARD = NCARD + 1
C       STORE AMAST
      WRITE (2) ((AMAST(MA,J),MA=1,MASTMX),J=1,NOVARS)
      WRITE (7,132) MASTMX,NOVARS,NDCARD
 132  FORMAT (6X,'DIMENSION AMAST(',I3,',',I3,')',T76,'D',I3)
      NDCARD = NDCARD + 1
      WRITE (7,134) NCARD
 134  FORMAT (6X,'READ (2) AMAST ',T77,I3)
      NCARD = NCARD + 1
C  STORE NJ
      WRITE (2) ((NJ(J,K),J=1,3),K=1,NUMTBL)
      WRITE (7,136) NUMTBL,NDCARD
 136  FORMAT (6X,'DIMENSION NJ( 3,',I3,')',T76,'D',I3)
      NDCARD = NDCARD + 1
      WRITE (7,138) NCARD
 138  FORMAT (6X,'READ (2) NJ',T77,I3)
      NCARD = NCARD + 1
C  STORE NK
      WRITE (2) ((NK(J,K),J=1,3),K=1,NUMTBL)
      WRITE (7,140) NUMTBL,NDCARD
 140  FORMAT (6X,'DIMENSION NK( 3,',I3,')',T76,'D',I3)
      NDCARD = NDCARD + 1
      WRITE (7,142) NCARD
 142  FORMAT (6X,'READ (2) NK',T77,I3)
```

```
      NCARD = NCARD + 1
C   STORE NL
      WRITE (2) ((NL(J,K),J=1,3),K=1,NUMTBL)
      WRITE (7,144) NUMTBL,NDCARD
 144  FORMAT (6X,'DIMENSION NL( 3,',I3,')',T76,'D',I3)
      NDCARD = NDCARD + 1
      WRITE (7,146) NCARD
 146  FORMAT (6X,'READ (2) NL',T77,I3)
      NCARD = NCARD + 1
 150  NCARD = NCARD -1
      NDCARD = NDCARD - 1
      WRITE (6,200) NDCARD,NCARD
 200  FORMAT ('1  ',I3,'  DIMENSION CARDS WERE PUNCHED.'///
     1 '   ',I3,'  READ CARDS WERE PUNCHED.')
      WRITE (6,210)
 210  FORMAT ('  NOTE ***   ON TSS, THESE CARD IMAGES ARE STORED IN THE
     1DATA SET ASSIGNED TO FT07F001 BY A DDEF CARD.'/'
     2SEE THE TABLE PROCESSOR USERS GUIDE.')
      ENDFILE 2
      STOP
      END
```

```
      SUBROUTINE TABCOM
C
C TABCOM IS THE PRIMARY SUBROUTINE FOR PROCESSING AERO TABLES.
C IT CONTROLS TABLE INPUT AND OUTPUT, DOES BREAKPOINT LIST COMPARISONS
C AND CATALOGING, AND STORAGE OF MOST AUXILIARY ARRAYS.
C
      COMMON VNAME,VARNM,TNAME,TNME,NVLGTH,BPSUBL,FUNC
     1,INEND,NOTABL
     2        ,JTN,NLENTH,NTABLE,BPL,NOVARS
     3,LISTDM,NCARD,NDCARD,IPUNCH,NUMTBL
      COMMON/TABLE/NTDIM,NL,NK,NJ
      DIMENSION NTDIM(200),NL(3,200),NK(3,200),NJ(3,200)
      REAL*8 VARNM,VNAME,TNME,TNAME
      DIMENSION VARNM(25) ,JTN(25), NLENTH(25,20),NTABLE(25,20)
     1,BPL(25,20,20),TNME(200)
      DIMENSION VNAME(3),NVLGTH(3),BPSUBL(25,3),FUNC(20,20,10)
      DATA JMAX/0/,KMAX/0/,LMAX/0/
      INEND=0                                                        6
      NOTABL = 0
      KLIM = 20
C NOVARS=NUMBER OF DIFFERENT VARIABLE NAMES                          7
 90   FORMAT(I5)                                                     8
      READ(5,90) NOVARS
      PRINT 91,NOVARS
 91   FORMAT(20H NUMBER OF VARIABLES,I5)
C VARNM=VARIABLE NAMES                                               10
      READ (5,100) (VARNM(J),J=1,NOVARS)
 100  FORMAT(10A8)                                                   12
      PRINT 101,(VARNM(J),J=1,NOVARS  )
 101  FORMAT(1H ,10A8)                                               1-1
C READ IN A TABLE                                                    13
 300  CALL TABIN                                                     14
C TEST FOR LAST TABLE INEND=1                                        15
      IF(INEND)1,1,199                                               16
 1    CONTINUE
      CALL TABOUT
C
C  STORE FUNC TABLE ON DISK + PUNCH RETRIEVAL CARDS
C  WHEN IPUNCH=0, DO NOT PUNCH
      IF (IPUNCH) 4,4,5
 5    NROW = NVLGTH(1)
      NCOL = NVLGTH(2)
      NPLN = NVLGTH(3)
      IF(LISTDM-2) 11,12,13
C  1D TABLE
 11   NCOL = NVLGTH(1)
```

```
      WRITE(2)(FUNC(1,K,1),K=1,NCOL)
      WRITE (7,801) TNAME,NCOL,NDCARD
  801 FORMAT(6X,'DIMENSION ',A8,'(',I3,')',T76,'D',I3)
      NDCARD = NDCARD + 1
      WRITE (7,804) TNAME,NCARD
  804 FORMAT (6X,'READ (2)',A8,T77,I3)
      NCARD = NCARD + 1
      GO TO 4
C  2D TABLE
 12   WRITE (2) ((FUNC(NR,NC,1),NR=1,NROW),NC=1,NCOL)
      WRITE (7,802) TNAME,NROW,NCOL,NDCARD
  802 FORMAT(6X,'DIMENSION ',A8,'(',I3,',',I3,')',T76,'D',I3)
      NDCARD = NDCARD + 1
      WRITE (7,805) TNAME,NCARD
 805  FORMAT (6X,'READ (2) ',A8,T77,I3)
      NCARD = NCARD + 1
      GO TO 4
C  3D TABLE
 13   WRITE (2) (((FUNC(NR,NC,NP),NR=1,NROW),NC=1,NCOL),NP=1,NPLN)
      WRITE (7,128) TNAME,NROW,NCOL,NPLN,NDCARD
 128  FORMAT (6X,'DIMENSION ',A8,'(',I3,',',I3,',',I3,')',T76,'D',I3)
      NDCARD = NDCARD + 1
      WRITE (7,130)  TNAME,NCARD
 130  FORMAT (6X,'READ (2) ',A8,T77,I3)
      NCARD = NCARD + 1
 4    CONTINUE
C STORE THE TABLE NAME
      TNME(NOTABL )=TNAME
C SAVE DIMENSION OF THE TABLE
      NTDIM(NOTABL)=LISTDM
C************************ SAVE **NEW** BREAK POINT LISTS *******************
 42   DO 40 K=1      ,LISTDM
      JFOUND=0
      DO 10 J=1,NOVARS
C FIND THE VARIABLE
      IF(VARNM(J)-VNAME(K))10,20,10
 20   JFOUND=1
C CHECK IF IDENTICAL BREAKPOINT LIST IS ALREADY SAVED
      JT=JTN(J)
C SEE IF BPSUBL HAS SAME LENGTH AS SOME ALREADY STORED LIST
C OF THE J TH VARIABLE
      DO 500 K1=1,JT
      IF(NVLGTH(K) .NE. NLENTH(J,K1)) GO TO 500
C SINCE LENGTH ARE EQUAL CHECK ELEMENT FOR ELEMENT
      NG=NLENTH(J,K1)
      DO 510 L=1,NG
```

```
      IF(BPL(J,K1,L) .NE. BPSUBL(L,K)) GO TO 500
 510  CONTINUE
C BP SUBLIST IS IDENTICAL TO THE  K1 TH LIST ALREADY STORED
      NK(K,NOTABL)=K1
      GO TO 520
 500  CONTINUE
C*********************************************************************
C LISTS ARE NOT EQUAL SINCE ONE ELEMENT IS DIFFERENT
C UPDATA BREAK POINT LIST COUNT FOR J TH VARIABLE                        31
 511  JTN(J)=JTN(J)+1                                                    32
C  CHECK FOR TOO MANY BP LISTS
      IF (JTN(J) .LE. KLIM) GO TO 512
      JTN(J) = JTN(J) - 1
      PRINT 80, VARNM(J),KLIM
 80   FORMAT (///'  THIS NEW BREAKPOINT LIST FOR   ',A8,'  WAS NOT SAVED
     1.'//'  THE MAXIMUM DIMENSIONED NUMBER OF',I4,'  BREAKPOINT LISTS H
     2AVE ALREADY BEEN SAVED FOR THIS VARIABLE.')
      GO TO 40
 512  JT = JTN(J)
C SAVE POSITION K OF J TH VARIABLE
      NK(K,NOTABL)=JT
C STORE LENGTH OF BREAKPOINT LIST                                        33
      NLENTH(J,JT     )=NVLGTH(K)
C STORE NUMBER OF THE TABLE                                              35
      NTABLE(J,JT)=NOTABL
C SAVE THE BREAKPOINT LIST                                               37
      NV=NVLGTH(K)
      DO 30 L=1,NV
      BPL(J,JT     ,L)=BPSUBL(L,K)
 30   CONTINUE                                                           39-4
C*********************************************************************
 520  CONTINUE
C THE FOLLOWING ARRAYS IDENTIFY J,  L IN BPL(J,K,L)
C SAVE NUMBER OF THE VARIABLE
      NJ(K,NOTABL)=J
C SAVE LENGTH OF THE BPL FOR VAR  1,2,3
      NL(K,NOTABL)=NVLGTH(K)
      GO TO 51
 10   CONTINUE                                                           40
 51   CONTINUE
C COMMENT IF THE VARIABLE IS NOT ON THE MASTER LIST                      41
C     PRINT 90,JFOUND                                                    41-1
      IF(JFOUND) 40,50,40                                                42
 50   PRINT 60,VNAME(K)                                                  43
 60   FORMAT(1H ,A8,18HNOT ON MASTER LIST)                               44
 40   CONTINUE                                                           45
```

```
C READ NEXT LIST                                                   46
      GO TO 300                                                    47
C SAVE COUNT OF TABLES READ
 199  NUMTBL = NOTABL
C PRINT OUT ALL BREAKPOINT LISTS                                   48
      DO 200 J=1,NOVARS
C PRINT VARIABLE NAME                                              50
      PRINT 210,VARNM(J)                                           51
 210  FORMAT(1H1,30X,A8)                                           52
C PRINT TABLE NUMBER,NAME,AND BREAKPOINT LIST                      53
      JT=JTN(J)
      DO 220 K=1,JT
      NX=NLENTH(J,K)
      NT=NTABLE(J,K)
 220  PRINT 230,NTABLE(J,K),TNME(NT),(BPL(J,K,L),L=1,NX)
 230  FORMAT(1H ,I4,1X,A8,8F10.5,3(/,14X,8F10.5))
 200  CONTINUE                                                     58
CCCCC
C      WRITE TABLE NUMBER, NAME, AND BREAKPOINT LISTS ON DISK
C       PUNCH RETRIEVAL CARDS
C  IF IPUNCH=0, DO NOT WRITE ON DISK OR PUNCH
      IF (IPUNCH) 250,250,260
 250  RETURN
C       FIND MAXIMUM SIZE OF EACH DIMENSION INDEX
 260  JMAX = NOVARS
      DO 700 J=1,NOVARS
      IF (JTN(J) .GT. KMAX) KMAX = JTN(J)
 700  CONTINUE
      DO 750 J=1,NOVARS
      JT = JTN(J)
      DO 750 K=1,JT
      IF (NLENTH(J,K) .GT. LMAX) LMAX = NLENTH(J,K)
 750  CONTINUE
      WRITE (2) NUMTBL
      WRITE (7,94 ) NCARD
 94   FORMAT (6X,'READ (2) NUMTBL',T77,I3)
      NCARD = NCARD + 1
C        STORE NOVARS
      WRITE (2) NOVARS
      WRITE (7,95) NCARD
 95   FORMAT (6X,'READ (2) NOVARS',T77,I3)
      NCARD = NCARD + 1
C        STORE VARNM
      WRITE (2) (VARNM,J=1,NOVARS)
      WRITE (7,105) NOVARS,NDCARD
 105  FORMAT (6X,'REAL*8 VARNM(',I3,')',T76,'D',I3)
```

```
      NDCARD = NDCARD + 1
      WRITE (7,106) NCARD
  106 FORMAT (6X,'READ (2) VARNM',T77,I3)
      NCARD = NCARD + 1
C        STORE BPL
      WRITE (2) (((BPL(J,K,L),J=1,JMAX),K=1,KMAX),L=1,LMAX)
      WRITE (7,124) JMAX,KMAX,LMAX,NDCARD
  124 FORMAT (6X,'DIMENSION BPL(',I3,',',I3,',',I3,')',T76,'D',I3)
      NDCARD = NDCARD + 1
      WRITE (7,126) NCARD
  126 FORMAT (6X,'READ (2) BPL',T77,I3)
      NCARD = NCARD + 1
C       STORE NLENTH
      WRITE (2) ((NLENTH(J,K),J=1,JMAX),K=1,KMAX)
      WRITE (7,110) JMAX,KMAX,NDCARD
  110 FORMAT (6X,'DIMENSION NLENTH(',I3,',',I3,')',T76,'D',I3)
      NDCARD = NDCARD + 1
      WRITE (7,112) NCARD
  112 FORMAT (6X,'READ (2) NLENTH',T77,I3)
      NCARD = NCARD + 1
C       STORE NTABLE
      WRITE (2) ((NTABLE(J,K),J=1,JMAX),K=1,KMAX)
      WRITE (7,114) JMAX,KMAX,NDCARD
  114 FORMAT (6X,'DIMENSION NTABLE(',I3,',',I3,')',T76,'D',I3)
      NDCARD = NDCARD + 1
      WRITE (7,116) NCARD
  116 FORMAT (6X,'READ (2) NTABLE',T77,I3)
      NCARD = NCARD + 1
C       STORE TNME
      WRITE (2) (TNME(N),N=1,NUMTBL)
      WRITE (7,118) NUMTBL,NDCARD
  118 FORMAT (6X,'REAL*8 TNME(',I3,')',T76,'D',I3)
      NDCARD = NDCARD + 1
      WRITE (7,119) NCARD
  119 FORMAT (6X,'READ (2) TNME',T77,I3)
      NCARD = NCARD + 1
C       STORE JTN
      WRITE (2) (JTN(J),J=1,NOVARS)
      WRITE (7,120) NOVARS,NDCARD
  120 FORMAT (6X,'DIMENSION JTN(',I3,')',T76,'D',I3)
      NDCARD = NDCARD + 1
      WRITE (7,122) NCARD
  122 FORMAT (6X,'READ (2) JTN',T77,I3)
      NCARD = NCARD + 1
C   STORE NTDIM
      WRITE (2) (NTDIM(N),N=1,NUMTBL)
```

```
      WRITE (7,132) NUMTBL,NDCARD
  132 FORMAT (6X,'DIMENSION NTDIM(',I3,')',T76,'D',I3)
      NDCARD = NDCARD + 1
      WRITE (7,134) NCARD
  134 FORMAT (6X,'READ (2) NTDIM',T77,I3)
      NCARD = NCARD + 1
      RETURN
      END
```

```
      SUBROUTINE TABIN
C TABIN READS A TABLE FROM CARDS
      COMMON VNAME,VARNM,TNAME,TNME,NVLGTH,BPSUBL,FUNC
     1,INEND,NOTABL                                                     63
     2      ,JTN,NLENTH,NTABLE,BPL,NOVARS
     3,LISTDM,NCARD,NDCARD
      REAL*8 VARNM,VNAME,TNME,TNAME
      DIMENSION VNAME(3),NVLGTH(3),BPSUBL(25,3),FUNC(20,20,10)
      DIMENSION ARG1(20),ARG2(20),ARG3(20)
      DIMENSION VARNM(25) ,JTN(25), NLENTH(25,20),NTABLE(25,20)
     1,BPL(25,20,20),TNME(200)
      REAL*8 COMENT
      DATA LLIM /20/
  240 READ (5,10,END=900,ERR=1200) COMENT,TNAME,(VNAME(N),N=1,3),
     1 (NVLGTH(J),J=1,3)
   10 FORMAT (A8,4A8,3I4)
      IFOUND = 0
C  CHECK FIRST CARD OF GROUP TO SEE IF IT IS A CORRECT HEADER CARD
C FIND LIST DIMENSION                                                   17
    7 LISTDM=0
      DO 2 J=1,3                                                        19
      IF(NVLGTH(J)-1)2,2,3                                              20
    3 LISTDM=LISTDM+1                                                   21
    2 CONTINUE                                                          22
      IF (LISTDM .EQ. 0) GO TO 260
C   CHECK IF TABLE TOO LARGE FOR ARRAY DIMENSIONS
      IF (NVLGTH(1) .GT. LLIM .OR. NVLGTH(2) .GT. LLIM .OR.
     1  NVLGTH(3) .GT. LLIM)  GO TO 1000
C   CHECK IF ALL VARIABLE NAMES ARE IN THE MASTER LIST
      DO 255 N=1,LISTDM
      DO 250 J=1,NOVARS
      IF (VNAME(N) .EQ. VARNM(J)) GO TO 252
  250 CONTINUE
      GO TO 255
  252 IFOUND = IFOUND + 1
  255 CONTINUE
      IF (IFOUND .EQ. LISTDM) GO TO 300
      WRITE (6,256)
  256 FORMAT ('   ALL OF VARIABLES NOT FOUND IN MASTER LIST.')
      GO TO 260
C   BAD HEADER CARD, PRINT AS READ + READ UNTIL FIND GOOD ONE
  260 WRITE (6,12) COMENT,TNAME,(VNAME(N),N=1,3)
   12 FORMAT ('   SEARCHING FOR A CORRECT TABLE HEADER CARD --- CARD POR
     1TION READ WAS',10X,A8,4A8)
      GO TO 240
  300 NOTABL = NOTABL + 1
```

```
      NROW=NVLGTH(1)
      NCOL=NVLGTH(2)
      NPLN=NVLGTH(3)
C                                                                       71
      IF(NROW-1)100,100,400
 100  READ (5,20,END=900,ERR=1220) (ARG1(K),FUNC(1,K,1),K=1,NCOL)
 20   FORMAT(2F8.3)                                                     75
      DO 810  K=1,20
 810  BPSUBL(K,1)=ARG1(K)
      NVLGTH(1)=NCOL
      NVLGTH(2)=1
      RETURN                                                            76
 400  DO 700 L=1,NPLN                                                   77
      IF(NPLN-1)43,200,43                                               78
 43   READ (5,40,END=900,ERR=1220) ARG3(L)
 200  READ (5,40,END=900,ERR=1220) (ARG1(J),J=1,NROW)
 40   FORMAT(8X,9F8.3)                                                  84
 500  DO 600 K=1,NCOL                                                   85
      READ (5,60,END=900,ERR=1220) ARG2(K),(FUNC(J,K,L),J=1,9)
      IF (NROW .GT. 9) READ (5,40,END=900,ERR=1220) (FUNC(J,K,L),
     1  J=10,NROW)
 600  CONTINUE
 60   FORMAT(10F8.3)                                                    88
 700  CONTINUE                                                          89
C STORING THE ARGUMENTS IN THE BREAK POINT SUBLIST                      90
      DO 800 K=1,25
      BPSUBL(K,1)=ARG1(K)                                               92
      BPSUBL(K,2)=ARG2(K)                                               93
 800  BPSUBL(K,3)=ARG3(K)                                               94
      RETURN                                                            95
 900  INEND=1                                                           96
      RETURN                                                            97
C   ERROR ROUTINE FOR OVERSIZE TABLES
 1000 WRITE (6,80) TNAME,(VNAME(N),N=1,3),(NVLGTH(J),J=1,3),LLIM
 80   FORMAT ('   TABLE  ',A8,'(',A8,A8,A8,')',' (',3I4,')   HAS A DIME
     1NSION LARGER THAN THE MAXIMUM ALLOWED SIZE OF ',I4)
      GO TO 240
C   READ ERROR ROUTINES
 1200 WRITE (6,1201)
 1201 FORMAT ('   READ ERROR OCCURRED WHILE TRYING TO READ A TABLE HEADE
     1R CARD.')
      GO TO 240
 1220 WRITE (6,1221)
 1221 FORMAT ('   READ ERROR OCCURRED WHILE TRYING TO READ AN ARGUMENT O
     1R FUNCTION CARD.')
      GO TO 240
```

98
END

```
      SUBROUTINE TABOUT
C         TABOUT PRINTS A COEFFICIENT TABLE                    WARNER 4/71
      COMMON VNAME,VARNM,TNAME,TNME,NVLGTH,BPSUBL,FUNC
     1,INEND,NOTABL
     2      ,JTN,NLENTH,NTABLE,BPL,NOVARS
     3,LISTDM,NCARD,NDCARD
      REAL*8 VARNM,VNAME,TNME,TNAME,DAT
      DIMENSION VNAME(3),NVLGTH(3),BPSUBL(25,3),FUNC(20,20,10)
      DIMENSION ARG1(20),ARG2(20),ARG3(20)
      DIMENSION VARNM(25) ,JTN(25), NLENTH(25,20),NTABLE(25,20)
     1,BPL(25,20,20),TNME(200)
      DATA JUMP/0/
C         GET DATE FIRST TIME THRU
      IF (JUMP) 200,100,200
  100 CALL DATE(DAT)
      JUMP = 1
C
  200 CONTINUE
      NROW = NVLGTH(1)
      NCOL = NVLGTH(2)
      NPLN = NVLGTH(3)
C         PRINT HEADER FOR CORRECT DIMENSIONAL TABLE
      IF (LISTDM-2) 300,500,600
C
C         ONE DIMENSIONAL HEADER + TABLE PRINT OUT
  300 WRITE (6,10) NOTABL,TNAME,VNAME(1),DAT
   10 FORMAT ('1',T8,'TABLE  ',I4,T60,A8,'( ',A8,')',T113,A8//)
      WRITE (6,20) VNAME(1),TNAME
   20 FORMAT ('0',T58,A8,8X,A8)
      NCOL = NVLGTH(1)
      WRITE (6,30)(BPSUBL(K,1),FUNC(1,K,1),K=1,NCOL)
   30 FORMAT ('0',T53,F13.6,F15.6)
      RETURN
C
C
C         TWO DIMENSIONAL HEADER
  500 WRITE (6,40) NOTABL,TNAME,VNAME(1),VNAME(2),DAT
   40 FORMAT ('1',T8,'TABLE  ',I4,T55,A8,'( ',A8,', ',A8,')',T113,A8//)
      GO TO 700
C
C         THREE DIMENSIONAL HEADER
  600 WRITE (6,50) NOTABL,TNAME,(VNAME(N),N=1,3),DAT
   50 FORMAT ('1',T8,'TABLE  ',I4,T50,A8,'( ',A8,', ',A8,', ',A8,')',
     $T113,A8//)
C
C         SET DO LOOP FOR PLANES OF 3D TABLE
```

```
  700 CONTINUE
      DO 1000 L=1,NPLN
C
C           IF TABLE 2D, DON'T PRINT ARG3 VALUE
      IF (NPLN .EQ. 1) GO TO 800
      IF (L .GT. 1) WRITE (6,98)
   98 FORMAT ('1')
      WRITE (6,60) VNAME(3),BPSUBL(L,3)
   60 FORMAT ('0',T60,A8,' =  ',G13.6//)
C
C          PRINT COLUMN ARGUMENT NAME + VALUES
  800 WRITE (6,70) VNAME(1),(BPSUBL(J,1),J=1,NROW)
   70 FORMAT ('0',T20,A8/(T14,9F13.6) )
      WRITE (6,99)
   99 FORMAT ('0')
C
C          PRINT ROW ARGUMENT NAME, VALUES AND TABLE FUNCTION VALUES
      WRITE (6,80) VNAME(2)
   80 FORMAT (' ',A8)
      DO 900 K=1,NCOL
  900 WRITE (6,90) BPSUBL(K,2),(FUNC(J,K,L),J=1,NROW)
   90 FORMAT ('0',F11.6,(T14,9F13.6))
 1000 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE NBLIST(AMAST,NMAST)
C NBLIST CONSTRUCTS THE NBPL CATALOG
      COMMON VNAME,VARNM,TNAME,TNME,NVLGTH,BPSUBL,FUNC
     1,INEND,NOTABL
     2      ,JTN,NLENTH,NTABLE,BPL,NOVARS
     3,LISTDM,NCARD,NDCARD,IPUNCH,NUMTBL
      REAL*8 VARNM,VNAME,TNME,TNAME
      DIMENSION VNAME(3),NVLGTH(3),BPSUBL(25,3),FUNC(20,20,10)
      DIMENSION VARNM(25) ,JTN(25), NLENTH(25,20),NTABLE(25,20)
     1,BPL(25,20,20),TNME(200)
      DIMENSION AMAST(35,25),NMAST(25),NBPL(25,35,20)
C TEMPORARY TABLE NAMES LIST FOR PRINT OUT ONLY
      DIMENSION TNMP(20)
      DO 200 J=1,NOVARS
      JT=JTN(J)
      DO 210 K=1,JT
      NBPL(J,1,K)=-1
      LM=1
 220  IF(BPL(J,K,1)-AMAST(LM,J))230,230,240
C BPL GT AMST FILL WITH -1 TO SHOW BELOW RANGE OF VARIABLE
 240  NBPL(J,LM,K)=-1
      LM=LM+1
      GO TO 220
C FIRST ELEMENT ON BPL = AMAST
 230  NBPL(J,LM,K)=1
      NSTART=LM+1
      L=2
      IENDL=0
      NM=NMAST(J)
      DO 250 LM=NSTART,NM
      IF(IENDL) 255,255,300
 255  IF(BPL(J,K,L)-AMAST(LM,J))270,270,260
C BPL GT AMAST
 260  NBPL(J,LM,K)=NBPL(J,LM-1,K)
      GO TO 250
C BPL EQ AMAST
 270  NBPL(J,LM,K)=NBPL(J,LM-1,K)+1
      IF(L-NLENTH(J,K))280,290,290
 280  L=L+1
      GO TO 250
 290  IENDL=1
      GO TO 250
C FILL WITH 0 S TO SHOW THAT ABOVE RANGE
 300  NBPL(J,LM,K)=0
 250  CONTINUE
 210  CONTINUE
```

```
C SET THE LAST TERM OF THE NBPL LIST = 0  UNLESS IT GOES TO LIMIT
C                 OF THE MASTER LIST
      LM=NMAST(J)
      DO 410 K=1,JT
      IF (NBPL(J,LM,K) .NE. 0) GO TO 410
      LN=LM+1
      DO 420 L= 1,LM
      LN=LN-1
      IF(NBPL(J,LN,K)) 430,420,430
  430 NBPL(J,LN,K)=0
      GO TO 410
  420 CONTINUE
  410 CONTINUE
  400 CONTINUE
C   WRITE OUT THE NBPL LISTS
      PRINT 305,VARNM(J)
  305 FORMAT(1H1,15HNBPL LISTS FOR ,A8)
      JT=JTN(J)
      DO 306 K=1,JT
      NT=NTABLE(J,K)
  306 TNMP(K)=TNME(NT)
      PRINT 330,(TNMP(K ),K=1,JT)
  330 FORMAT(14X,15A6)
      JT=JTN(J)
      NM=NMAST(J)
      DO 320 L=1,NM
  320 PRINT 310,L,AMAST(L,J),(NBPL(J,L,K),K=1,JT)
  310 FORMAT(I5,F8.3,15I6,(/13X,15I6))
  200 CONTINUE
C  STORE NBPL ON DISK + PUNCH RETRIEVAL CARDS IF IPUNCH = 1
      IF (IPUNCH) 600,600,500
C  FIND MAX SIZE OF EACH DIMENSION
  500 KMAX = 0
      NMMAX = 0
      DO 520 J=1,NOVARS
      IF (JTN(J) .GT. KMAX) KMAX = JTN(J)
      IF (NMAST(J) .GT. NMMAX) NMMAX = NMAST(J)
  520 CONTINUE
      WRITE (2) (((NBPL(J,L,K),J=1,NOVARS),L=1,NMMAX),K=1,KMAX)
      WRITE (7,550) NOVARS,NMMAX,KMAX,NDCARD
  550 FORMAT (6X,'DIMENSION NBPL(',I3,',',I3,',',I3,')',T76,'D',I3)
      NDCARD = NDCARD + 1
      WRITE (7,570) NCARD
  570 FORMAT (6X,'READ (2) NBPL',T77,I3)
      NCARD = NCARD + 1
  600 CONTINUE
```

```
RETURN
END
```

```
      SUBROUTINE MLIST(TEMP,JS,ALPHA,NTOTAL)
C MLIST CONSTRUCTS A MASTER LIST OF BREAKPOINT VALUES FOR A VARIABLE
      DIMENSION ALPHA(1),TEMP(1)
      JEND=JS
      DO 1 I=1,JEND
 1    ALPHA(I)=TEMP(I)
      JNEW=1
      ISTART=1
      IEND=JEND-1
 6    JEND=JEND-1
      JSTART=JNEW
      DO 13 J=JSTART,JEND
      JNEW=J
 11   DO 12 I=ISTART,IEND
      INEW=I
      IF( ALPHA(I+1)-ALPHA(I)-0.00001)20,20,12
 20   IF(ABS(ALPHA(I+1)-ALPHA(I))-0.00001)14,21,21
 21   TEMPA=ALPHA(I)
      ALPHA(I)=ALPHA(I+1)
      ALPHA(I+1)=TEMPA
 12   CONTINUE
      ISTART=1
      GO TO 13
 14   DO 15 K=INEW,IEND
 15   ALPHA(K)=ALPHA(K+1)
      ISTART=INEW
      IEND=IEND-1
      IF(ISTART-IEND) 6,22,22
 22   ISTART=1
      JNEW=JNEW+1
      GO TO 6
 13   CONTINUE
      NTOTAL=JEND+1
      RETURN
      END
```

APPENDIX II

PLOTTING PROGRAM LISTINGS

Data verification by plotting with the S-C 4020 plotter is done using the following programs:

1. Main
2. PLOT1D
3. PLOT23
4. SCLIM
5. FLIP2
6. FLP213
7. FLP321
8. FLP132
9. FLP231
10. FLP312

The subroutines PLOT1D and PLOT23 make use of subroutines in the Ames package of programs for the S-C 4020 plotter.

```
C  MAIN PROGRAM FOR PLOTTING AERO TABLES STORED ON DISK BY NEUMAN METHOD
      COMMON /SCALE/ CMIN,CMAX,V1MIN,V1MAX
      DIMENSION C(8000),FC(8000),V1(20),V2(20),V3(20)
      REAL*8 VAR1,VAR2,VAR3,TNAM
      DATA IORDER/0/, LTABL/0/
      LOGICAL FFLAG/.TRUE./
C
C ***   USER'S DIMENSION AND READ CARDS GO HERE    ***
C ***      IN PLACE OF THESE    ***
      DIMENSION CM       (  3)
      DIMENSION CLDA     (  3,  4,  2)
      DIMENSION CLMN     (  3,  3)
      REAL*8 VARNM(  3)
      DIMENSION BPL(  3,  2,  4)
      DIMENSION NLENTH(  3,  2)
      DIMENSION NTABLE(  3,  2)
      REAL*8 TNME(  3)
      DIMENSION JTN(  3)
      DIMENSION NTDIM(  3)
      DIMENSION NBPL(  3,  7,  2)
      DIMENSION NMAST (  3)
      DIMENSION AMAST(  7,  3)
      DIMENSION NJ( 3,  3)
      DIMENSION NK( 3,  3)
      DIMENSION NL( 3,  3)
      READ (2)CM
      READ (2) CLDA
      READ (2) CLMN
      READ (2) NUMTBL
      READ (2) NOVARS
      READ (2) VARNM
      READ (2) BPL
      READ (2) NLENTH
      READ (2) NTABLE
      READ (2) TNME
      READ (2) JTN
      READ (2) NTDIM
      READ (2) NBPL
      READ (2) NMAST
      READ (2) AMAST
      READ (2) NJ
      READ (2) NK
      READ (2) NL
C
C  INITIALIZE PLOTTER PACKAGE
      CALL READIN
```

```
      CALL BIGV
      CALL CAMRAV(9)
C
C        REWIND TO START OF DATASET & READ TO FIRST TABLE
      REWIND 2
C
C  START LOOP TO PLOT TABLES WHEN IORDER = -1
      NTAB = NUMTBL
  100 DO 1000 NPLOT=1,NTAB
 10   IF (IORDER) 120,140,140
 120  NOTABL = NPLOT
      GO TO 160
C        READ CARD FOR IORDER, TABLE NO. , & VARIABLE NAMES
 140  READ (5,20,END=1020,ERR  =920) IORDER,NOTABL,VAR1,VAR2,VAR3
     1 ,CMIN,CMAX,V1MIN,V1MAX
 20   FORMAT (2I8,3A8,4F8.4)
C        WHEN IORDER = -1, NOTABL BLANK = 0, SO START DO LOOP
      IF (NOTABL) 160,100,160
C
C        READ IN CORRECT TABLE ---  LTABL = LAST TABLE NO. READ IN
 160  IF (NOTABL - LTABL) 180,280,200
C        RESET TO FIRST TABLE
 180  REWIND 2
      LTABL = 0
      GO TO 160
C       READ NEXT TABLE
 200  LTABL = LTABL + 1
      IF (LTABL - NUMTBL) 210,210,800
 210  NTD = NTDIM(LTABL)
      GO TO (220,240,260), NTD
C         1D TABLE SIZE
 220  NL1 = NL(1,LTABL)
      NC = NL1
      GO TO 270
C        2D TABLE SIZE
 240  NL1 = NL(1,LTABL)
      NL2 = NL(2,LTABL)
      NC = NL1 * NL2
      GO TO 270
C   3D TABLE SIZE
 260  NL1 = NL(1,LTABL)
      NL2 = NL(2,LTABL)
      NL3 = NL(3,LTABL)
      NC = NL1 * NL2 * NL3
C        READ TABLE INTO 1 DIMENSIONAL ARRAY C
 270  READ (2,ERR=950) (C(LN),LN=1,NC)
```

```
      FFLAG = .TRUE.
      GO TO 160
 280  CONTINUE
C  SEE IF TABLE IN C IS FLIPPED;  IF SO, REREAD
C          FFLAG = .FALSE. INDICATES TABLE WAS FLIPPED
      IF (FFLAG) GO TO 290
      BACKSPACE 2
      GO TO 270
 290  CONTINUE
C
C  GET TABLE NO. OF DIMENSIONS & GO TO PROPER ROUTINE
      NTD = NTDIM(NOTABL)
      GO TO (300,400,500), NTD
C  1D TABLE --- MOVE ARG & FUNC LISTS TO TEMPORARY LISTS & PLOT.
C   NO CHOICE OF PLOTTING ORDER
 300  TNAM = TNME(NOTABL)
C  GET LOCATION OF ARG LIST IN BPL
      NJ1 = NJ(1,NOTABL)
      NK1 = NK(1,NOTABL)
      VAR1 = VARNM(NJ1)
      DO 340 L=1,NL1
 340  V1(L) = BPL(NJ1,NK1,L)
      CALL PLOT1D(V1,C ,NL1,VAR1,TNAM,NOTABL)
      GO TO 800
C  2D TABLE --- MOVE INFO TO TEMPORARY LISTS & PLOT IN CORRECT ORDER
 400  TNAM = TNME(NOTABL)
      NJ1 = NJ(1,NOTABL)
      NJ2 = NJ(2,NOTABL)
      NK1 = NK(1,NOTABL)
      NK2 = NK(2,NOTABL)
      DO 420 L1=1,NL1
 420  V1(L1) = BPL(NJ1,NK1,L1)
      DO 422 L2=1,NL2
 422  V2(L2) = BPL(NJ2,NK2,L2)
C  IF IORDER=1, DETERMINE PLOTTING ORDER & SWITCH LISTS
      IF (IORDER) 428,428,440
C   VARS IN CORRECT ORDER, SO PLOT.
 428  VAR1 = VARNM(NJ1)
      VAR2 = VARNM(NJ2)
 430  CALL PLOT2D(V1,V2,C ,NL1,NL2,VAR1,VAR2,TNAM,NOTABL)
      GO TO 800
C   COMPARE VAR NAMES
 440  IF (VAR1 .NE. VARNM(NJ1)) GO TO 460
      IF (VAR2 .NE. VARNM(NJ2)) GO TO 900
      GO TO 430
C   VARS NEED SWITCHED, DOUBLE CHECK & DO IT
```

```
 460  IF (VAR1 .NE. VARNM(NJ2)) GO TO 900
      IF (VAR2 .NE. VARNM(NJ1)) GO TO 900
C   FLIP TABLE
      CALL FLIP2(C ,NL1,NL2,FC )
      FFLAG = .FALSE.
C   PLOT THE TABLE
 480  CALL PLOT2D(V2,V1,FC ,NL2,NL1,VAR1,VAR2,TNAM,NOTABL)
      GO TO 800
C  3D TABLE
 500  TNAM = TNME(NOTABL)
      NJ1 = NJ(1,NOTABL)
      NJ2 = NJ(2,NOTABL)
      NJ3 = NJ(3,NOTABL)
      NK1 = NK(1,NOTABL)
      NK2 = NK(2,NOTABL)
      NK3 = NK(3,NOTABL)
      DO 520 L1=1,NL1
 520  V1(L1) = BPL(NJ1,NK1,L1)
      DO 522 L2=1,NL2
 522  V2(L2) = BPL(NJ2,NK2,L2)
      DO 524 L3=1,NL3
 524  V3(L3) = BPL(NJ3,NK3,L3)
C  IF IORDER=1, DETERMINE PLOTTING ORDER & SWITCH LISTS
      IF (IORDER) 540,540,530
C   COMPARE VAR NAMES
 530  IF (VAR1 .NE. VARNM(NJ1)) GO TO 550
      IF (VAR2 .NE. VARNM(NJ2)) GO TO 570
      IF (VAR3 .NE. VARNM(NJ3)) GO TO 900
C   PLOT IN ORIGINAL ORDER
 540  VAR1 = VARNM(NJ1)
      VAR2 = VARNM(NJ2)
      VAR3 = VARNM(NJ3)
      CALL PLOT3D(V1,V2,V3,C ,NL1,NL2,NL3,VAR1,VAR2,VAR3,TNAM,NOTABL)
      GO TO 800
C
 550  IF (VAR1 .NE. VARNM(NJ2)) GO TO 560
      IF (VAR2 .NE. VARNM(NJ1)) GO TO 580
      IF (VAR3 .NE. VARNM(NJ3)) GO TO 900
C   FLIP TABLE TO 213 ORDER & PLOT
      CALL FLP213(C ,NL1,NL2,NL3,FC )
      FFLAG = .FALSE.
      CALL PLOT3D(V2,V1,V3,FC ,NL2,NL1,NL3,VAR1,VAR2,VAR3,TNAM,NOTABL)
      GO TO 800
C
 560  IF (VAR1 .NE. VARNM(NJ3)) GO TO 900
      IF (VAR2 .NE. VARNM(NJ2)) GO TO 590
```

```
      IF (VAR3 .NE. VARNM(NJ1)) GO TO 900
C   FLIP TABLE TO 321 ORDER & PLOT
      CALL FLP321(C ,NL1,NL2,NL3,FC )
      FFLAG = .FALSE.
      CALL PLOT3D(V3,V2,V1,FC ,NL3,NL2,NL1,VAR1,VAR2,VAR3,TNAM,NOTABL)
      GO TO 800
C
 570  IF (VAR2 .NE. VARNM(NJ3)) GO TO 900
      IF (VAR3 .NE. VARNM(NJ2)) GO TO 900
C   FLIP TABLE TO 132 ORDER & PLOT
      CALL FLP132(C ,NL1,NL2,NL3,FC )
      FFLAG = .FALSE.
      CALL PLOT3D(V1,V3,V2,FC ,NL1,NL3,NL2,VAR1,VAR2,VAR3,TNAM,NOTABL)
      GO TO 800
C
 580  IF (VAR2 .NE. VARNM(NJ3)) GO TO 900
      IF (VAR3 .NE. VARNM(NJ1)) GO TO 900
C   FLIP TABLE TO 231 ORDER & PLOT
      CALL FLP231(C ,NL1,NL2,NL3,FC )
      FFLAG = .FALSE.
      CALL PLOT3D(V2,V3,V1,FC ,NL2,NL3,NL1,VAR1,VAR2,VAR3,TNAM,NOTABL)
      GO TO 800
C
 590  IF (VAR2 .NE. VARNM(NJ1)) GO TO 900
      IF (VAR3 .NE. VARNM(NJ2)) GO TO 900
C   FLIP TABLE TO 312 ORDER & PLOT
      CALL FLP312(C ,NL1,NL2,NL3,FC )
      FFLAG = .FALSE.
      CALL PLOT3D(V3,V1,V2,FC ,NL3,NL1,NL2,VAR1,VAR2,VAR3,TNAM,NOTABL)
  800 CONTINUE
      WRITE (6,810) IORDER,NOTABL
 810  FORMAT (' IORDER = ',I3,'     TABLE',I3,'  PLOTTED')
      IF (IORDER) 1000,140,140
C
C
C
C  ERROR ROUTINE FOR INCORRECT VARIABLE NAMES ON PLOT ORDER CARD.
 900  WRITE (6,901) NOTABL,VAR1,VAR2,VAR3,VARNM(NJ1),VARNM(NJ2),
     1 VARNM(NJ3)
 901  FORMAT(//'   INCORRECT VARIABLE NAME ON CARD TO PLOT TABLE ',I3,/
     1'      VARIABLE NAMES ON CARD ARE   ',A8,5X,A8,5X,A8,/
     2'      CORRECT NAMES AS STORED ARE  ',A8,5X,A8,5X,A8,//)
      GO TO 140
C  PLOT CARD READ ERROR
 920  WRITE (6,921) IORDER,NOTABL,VAR1,VAR2,VAR3
 921  FORMAT (//'        PLOT CARD READ ERROR',/
```

```
     1 '        CARD HAS IORDER = ',I3,' ,  NOTABL = ',I3,' AND VARIABLE N
     2AMES = ',3A8//)
      GO TO 140
C   DISK READ ERROR
 950  WRITE (6,951) LTABL,NTDIM(LTABL),NL1,NL2,NL3
 951  FORMAT (//'  DISK READ ERROR',/'       TRYING TO READ TABLE NO.'
     1 ,I3,' WITH ',I3,' DIMENSIONS'/'      STORED SIZE IS ',3I5//)
      GO TO 800
 1000 CONTINUE
 1020 CONTINUE
      CALL EOFTV
      REWIND 1
      STOP
      END
```

```
      SUBROUTINE PLOT1D(V1,C,NV1,MTITLE,ITITLE,NOTABL)
C  THIS SUBROUTINE PLOTS A ONE DIMENSIONAL TABLE.
      DIMENSION V1(1),C(1),MTITLE(2),ITITLE(2)
      DIMENSION TITLE(80),TEMP(20)
      DIMENSION TTITLE(3)
C         V1   VARIABLE 1
C          C    AERO COEFF
C        NV1         NUMBER OF VARIABLE 1'S
C        NC          NUMBER OF C      TABLES
C         MTITLE     NAME OF VARIABLE 1
C         ITITLE     NAME OF AERO COEFF
C          TTITLE        TABLE NO. LABEL
      LOGICAL IFLAGX,IFLAGY
      DATA IFLAGX,IFLAGY/.TRUE.,.TRUE./
      DATA DCX,DCY/8.0,8.0/
      DATA TTITLE/'  TA','BLE ','    '/
C
C
C        NAND       NUMBER OF POINTS PER GRAPH
      NAND=NV1
      JOLD=0
      IAPT=0
      NC=NV1
      CALL SCLIM(C,NC,V1,NV1,XL,XR,YL,YR)
      CALL DXDYV(1,XL,XR,DX,NPT,IPT,NX,DCX,IERRX)
   14 CALL DXDYV(2,YL,YR,DY,MPT,JPT,NY,DCY,IERRY)
      IF(NX .GE. 6)   NX=6
      IF(NY .GE. 6)   NY=6
      IF(IERRX .EQ. 0)  GO TO 15
      WRITE(6,800)
  800 FORMAT(49H IERRX GREATER THAN ZERO. SEE CALL DXDYV IN PLOT2)
      IFLAGX=.FALSE.
   15 IF(IERRY .EQ. 0)  GO TO 20
      WRITE(6,801)
  801 FORMAT(49H IERRY GREATER THAN ZERO. SEE CALL DXDYV IN PLOT2)
      IFLAGY=.FALSE.
   20 IF(.NOT. IFLAGX .AND. .NOT. IFLAGY) GO TO 999
C        DRAW GRID
C
      CALL GRID1V(1,XL,XR,YL,YR,DX,DY,NPT,MPT,IPT,JPT,NX,NY)
C
C        PRINT OUT VARIABLE 1
C
      CALL RITE2V(480,15,1000,90,2,8,1,MTITLE,NLAST2)
C        PRINT OUT AERO COEFF
      CALL RITE2V(20,480,1000,180,2,8,1,ITITLE,NLASTA)
```

```
C           CONVERT & PRINT OUT TABLE NO.
      CALL CVRT(NOTABL,1,'(I4)',ATABL,1,'(A4)')
      TTITLE(3) = ATABL
      CALL RITE2V(20,50,400,180,2,12,1,TTITLE,NLASTZ)
C        START PLOTTING
      CALL APLOTV(NAND,V1,C    ,1,1,1,38,IAPT)
      CALL APLOTV(NAND,V1,C    ,1,1,1,38,IAPT)
C
C        CONNECT THE POINTS
      NT1=NV1-1
      DO 210 II=1,NT1
      IX1=NXV(V1(II))
      IX2=NXV(V1(II+1))
      IY1 = NYV(C(II))
      IY2 = NYV(C(II+1))
      CALL LINEV(IX1,IY1,IX2,IY2)
      CALL LINEV(IX1,IY1,IX2,IY2)
  210 CONTINUE
C
C        ERROR COUNT ON POINTS NOT PLOTTED
C
      IF(IAPT .GT. 3) WRITE(6,850) IAPT
  850 FORMAT(36H THE NUMBER OF POINTS NOT PLOTTED IS,I5)
C
C
  999 RETURN
      END
```

```
      SUBROUTINE PLOT23(V1,V2,V3,CIN,NV1,NV2,NV3,MTITLE,NTITLE,KTITLE,
     1 ITITLE,NOTABL)
C
C  THIS SUBROUTINE PLOTS EITHER 2 OR 3 DIMENSIONAL GRAPHS DEPENDING ON
C     THE ENTRY POINT USED.
C
      DIMENSION V1(1),V2(1),V3(1),TEMP(5000),MTITLE(2),NTITLE(2),
     1           KTITLE(2),ITITLE(2),MARK(14),CIN(1),C(8000)
      DIMENSION DICTY(80)
      DIMENSION TITLE1(80)
      DIMENSION TTITLE(3)
C
C          V1   VARIABLE 1
C          V2   VARIABLE 2
C          V3   VARIABLE 3
C
C         NV1                    VARIABLE 1'S
C         NV2                    VARIABLE 2'S
C         NV3        NUMBER OF VARIABLE 3'S
C         NC                     C      TABLES
C          C     AERO COEFF
C          MTITLE    NAME OF VARIABLE 1
C          NTITLE    NAME OF VARIABLE 2
C          KTITLE    NAME OF VARIABLE 3
C          ITITLE    NAME OF AERO COEFF
C           TTITLE        TABLE NO. LABEL
C
      LOGICAL IFLAGX,IFLAGY
      DATA IFLAGX,IFLAGY/.TRUE.,.TRUE./
      DATA DCX,DCY/8.0,8.0/
      DATA TTITLE/'  TA','BLE ','    '/
C
      DATA MARK/38,52,55,44,54,63,19,24,16,0,57,42,36,20/
C
      RETURN
C
C
      ENTRY PLOT2D(V1,V2,CIN,NV1,NV2,MTITLE,NTITLE,ITITLE,NOTABL)
      NV3 = 1
      GO TO 5
      ENTRY PLOT3D(V1,V2,V3,CIN,NV1,NV2,NV3,MTITLE,NTITLE,KTITLE,ITITLE,
     1 NOTABL)
C        NAND    NUMBER OF POINTS PER GRAPH
    5 NAND = NV1 * NV2
C
      ISH=0
```

```
      ISF=0
      IAPT=0
      IEND=0
      ISTART=1
C
      NC = NV1 * NV2 * NV3
      DO 13 I=1,NC
 13   C(I) = CIN(I)
      CALL SCLIM(C,NC,V1,NV1,XL,XR,YL,YR)
      CALL DXDYV(1,XL,XR,DX,NPT,IPT,NX,DCX,IERRX)
   14 CALL DXDYV(2,YL,YR,DY,MPT,JPT,NY,DCY,IERRY)
      IF(NX .GT. 6)    NX=6
      IF(NY .GT. 6)    NY=6
      IF(IERRX .EQ. 0)  GO TO 15
      WRITE(6,800)
  800 FORMAT(49H IERRX GREATER THAN ZERO. SEE CALL DXDYV IN PLOT3)
      IFLAGX=.FALSE.
   15 IF(IERRY .EQ. 0)  GO TO 20
      WRITE(6,801)
  801 FORMAT(49H IERRY GREATER THAN ZERO. SEE CALL DXDYV IN PLOT3)
      IFLAGY=.FALSE.
   20 IF(.NOT. IFLAGX .AND. .NOT. IFLAGY) GO TO 999
      IF (NV3 .LE. 1) GO TO 310
      NT3=2*NV3
      CALL CVRT(V3,NV3,'(20F8.2)',TITLE1,NT3,'(20(A4,A4))')
 310  CONTINUE
      NT2 = 2 * NV2
      CALL CVRT(V2,NV2,'(20F8.2)',DICTY,NT2,'(20(A4,A4))')
      CALL CVRT(NOTABL,1,'(I4)',ATABL,1,'(A4)')
      TTITLE(3) = ATABL
C
C        START LOOPS
      M1=1
      M2=NAND
C
      DO 400  M=1,NV3
      MNEW=M
C        DRAW GRID
C
      CALL GRID1V(1,XL,XR,YL,YR,DX,DY,NPT,MPT,IPT,JPT,NX,NY)
C
C        PRINT OUT VARIABLE 1,2,3
C
      CALL RITE2V(500,15,1000,90,2,8,1,MTITLE,NLASTM)
  251 IQX=775
      IQY=950
```

```
260 IQM=IQX+24
    IQXM=IQX
    IQYM=IQY
    IQX=IQX+20
    IQD=IQX+144
    NDICTY=8*NV2
    CALL RITE2V(IQX,IQY,IQD,90,2,NDICTY,1,DICTY,NLASTD)
    IQX=IQX-10
    IQY=IQY+20
    CALL RITE2V(IQX,IQY,1000,90,2,8,1,NTITLE,NLASTN)
    IF (NV3 .LE. 1) GO TO 350
    CALL RITE2V(90,1000,1000,90,2,8,1,KTITLE,NLASTK)
350 CONTINUE
    CALL RITE2V(20,480,1000,180,2,8,1,ITITLE,NLASTA)
    CALL RITE2V(20,50,400,180,2,12,1,TTITLE,NLASTZ)
    IF (NV3 .LE. 1) GO TO 370
    MCAL=2*MNEW-1
    CALL RITE2V(200,1000,1000,90,2,5,4,TITLE1(MCAL),NLASTT)
C     SET THE STARTING ADDRESS OF BOTH ARRAYS TO 1
370 DO 261  N=1,NV2
    CALL PLOTV(IQXM,IQYM,MARK(N))
    CALL PLOTV(IQXM,IQYM,MARK(N))
261 IQYM=IQYM-25
    IF (NV3 .LE. 1) GO TO 199
    DO 65 J=1,NAND
    JNEW=J
    ISC=ISH+JNEW
 65 C(J)=C(ISC)
    ISH=ISC
199 CONTINUE
C        START PLOTTING
    DO 200  I=1,NV1
    INEW=I
    CALL APLOTV(NAND,V1(I),C(I),0,NV1,NV2,MARK,IAPT)
    CALL APLOTV(NAND,V1(I),C(I),0,NV1,NV2,MARK,IAPT)
    CALL APLOTV(NAND,V1(I),C(I),0,NV1,NV2,MARK,IAPT)
200 CONTINUE
C
C        CONNECT THE POINTS
    JSTART=ISTART
    NT1=NV1-1
    DO 210 II=1,NT1
    IX1=NXV(V1(II))
    IX2=NXV(V1(II+1))
    DO 205 JJ=JSTART,NAND,NV1
    IY1=NYV(C(JJ))
```

```
      IY2=NYV(C(JJ+1))
      CALL LINEV(IX1,IY1,IX2,IY2)
      CALL LINEV(IX1,IY1,IX2,IY2)
  205 CONTINUE
C
  210 JSTART=JSTART+1
C        ERROR COUNT ON POINTS NOT PLOTTED
C
      IF(IAPT .GT. 3) WRITE(6,850) IAPT,V3(M)
  850 FORMAT(36H THE NUMBER OF POINTS NOT PLOTTED IS,I5,10H FOR  V3 =,
     1       F5.2)
C
  400 CONTINUE
C
  999 RETURN
      END
```

```
      SUBROUTINE SCLIM(C,NC,V1,NV1,XL,XR,YL,YR)
C
C  SET SCALING LIMITS TO THOSE READ ON PLOT CARD OR TO TABLE LIMITS IF NOT READ
C  IF EITHER LIMIT SET ON PLOT CARD, THEN OTHER CORRESPONDING LIMIT
C        MUST ALSO BE SET FOR PREDICTABLE RESULTS.
C
      COMMON /SCALE/ CMIN,CMAX,V1MIN,V1MAX
      DIMENSION C(1),V1(1)
C
C    VAR SCALING
      IF (V1MIN) 620,600,620
  600 IF (V1MAX) 620,610,620
  610 XL = V1(1)
      XR = V1(NV1)
      GO TO 700
  620 XL = V1MIN
      XR = V1MAX
C
C    FUNC SCALING
  700 IF (CMIN) 730,710,730
  710 IF (CMAX) 730,720,730
  720 YL = C(1)
      YR = C(1)
      DO 10 J=2,NC
      YL = AMIN1(YL,C(J))
   10 YR = AMAX1(YR,C(J))
      RETURN
  730 YL = CMIN
      YR = CMAX
      RETURN
      END
```

```
      SUBROUTINE FLIP2(A,N,M,B)
C FLIP2 REARRANGES A 2 DIMENSIONAL ARRAY FOR PLOTTING
      DIMENSION A(N,M)  ,B(M,N)
      DO 1 I=1,N
      DO 1 J=1,M
    1 B(J,I) =A(I,J)
      RETURN
      END


      SUBROUTINE FLP213(A,L,M,N,B)
C FLP  SUBROUTINES REARRANGE 3 DIMENSIONAL ARRAYS FOR PLOTTING
      DIMENSION A(L,M,N),B(M,L,N)
      DO 100 I=1,L
      DO 100 J=1,M
      DO 100 K=1,N
  100 B(J,I,K) = A(I,J,K)
      RETURN
      END


      SUBROUTINE FLP321(A,L,M,N,B)
      DIMENSION A(L,M,N),B(N,M,L)
      DO 100 I=1,L
      DO 100 J=1,M
      DO 100 K=1,N
  100 B(K,J,I) = A(I,J,K)
      RETURN
      END


      SUBROUTINE FLP132(A,L,M,N,B)
      DIMENSION A(L,M,N),B(L,N,M)
      DO 100 I=1,L
      DO 100 J=1,M
      DO 100 K=1,N
  100 B(I,K,J) = A(I,J,K)
      RETURN
      END
```

```
      SUBROUTINE FLP231(A,L,M,N,B)
      DIMENSION A(L,M,N),B(M,N,L)
      DO 100 I=1,L
      DO 100 J=1,M
      DO 100 K=1,N
100   B(J,K,I) = A(I,J,K)
      RETURN
      END
```

```
      SUBROUTINE FLP312(A,L,M,N,B)
      DIMENSION A(L,M,N),B(N,L,M)
      DO 100 I=1,L
      DO 100 J=1,M
      DO 100 K=1,N
100   B(K,I,J) = A(I,J,K)
      RETURN
      END
```

APPENDIX III

EAI 8400 CONVERSION PROGRAM LISTINGS

The following programs convert the tables to the proper format for input to the EAI 8400:

1. Main
2. FTOB

```
C   MAIN PROGRAM FOR PUNCHING EAI8400 CARDS
      DIMENSION F   (20,20,10),      NOLIST(57)
     1,IB(10)    ,M(17)
C*****************************************************************************
C
C DATA DEPENDENT DIMENSION AND READ STATEMENTS
C OBTAINED FROM THE TABLE PRE-PROCESSOR
C    REPLACE THESE
      DIMENSION CM       (  3)                                           D  1
      DIMENSION CLDA     (  3,  4,  2)                                   D  2
      DIMENSION CLMN     (  3,  3)                                       D  3
      REAL*8 VARNM(  3)                                                  D  4
      DIMENSION BPL(  3,  2,  4)                                         D  5
      DIMENSION NLENTH(  3,  2)                                          D  6
      DIMENSION NTABLE(  3,  2)                                          D  7
      REAL*8 TNME(  3)                                                   D  8
      DIMENSION JTN(  3)                                                 D  9
      DIMENSION NTDIM(  3)                                               D 10
      DIMENSION NBPL(  3,  7,  2)                                        D 11
      DIMENSION NMAST (  3)                                              D 12
      DIMENSION AMAST(  7,  3)                                           D 13
      DIMENSION NJ( 3,  3)                                               D 14
      DIMENSION NK( 3,  3)                                               D 15
      DIMENSION NL( 3,  3)                                               D 16
      READ (2)CM                                                            1
      READ (2) CLDA                                                         2
      READ (2) CLMN                                                         3
      READ (2) NUMTBL                                                       4
      READ (2) NOVARS                                                       5
      READ (2) VARNM                                                        6
      READ (2) BPL                                                          7
      READ (2) NLENTH                                                       8
      READ (2) NTABLE                                                       9
      READ (2) TNME                                                        10
      READ (2) JTN                                                         11
      READ (2) NTDIM                                                       12
      READ (2) NBPL                                                        13
      READ (2) NMAST                                                       14
      READ (2) AMAST                                                       15
      READ (2) NJ                                                          16
      READ (2) NK                                                          17
      READ (2) NL                                                          18
C
C*****************************************************************************
C
      DATA M/'1','2','3','4','5','6','7','8','9','0','-',',','.',' ','/'
```

```
     1,'(',')'/
C
      PRINT 2000
      PRINT 2001
      PRINT 1999
      PRINT 2002
      PRINT 2003
      PRINT 2005
      PRINT 2006
      PRINT 2007
      PRINT 2008
 2000 FORMAT(1H1,'NONE OF THE BREAKPOINTLISTS PUNCHED IS DUPLICATED')
 2001 FORMAT(1H ,'AS INITIALLY PUNCHED ALL BREAKPOINTLISTS ARE CALLED BY THE
     1 THE SAME NAME,')
 1999 FORMAT(1H ,' SINCE WE CANNOT PREDICT WHAT THE NAMES OF THE INDIVID
     1UAL BREAKPONTLISTS WILL BE.')
 2002 FORMAT(1H ,'SUGGESTION=CHANGE CARDS AS PER EXAMPLE')
 2003 FORMAT(1H ,'FROM=     THRUPUT FAFB(CJA      ,DA      )')
 2005 FORMAT(1H ,'TO  =     THRUPUT FAFB(CJA1     ,DA1     )')
 2006 FORMAT(1H ,'ACCORDING TO THE NUMBER OF THE BREAKPOINTLIST FOR THE
     1 GIVEN TABLE.'/' THIS NUMBER IS GIVEN IN THE COMMENT LINE.')
 2007 FORMAT(1H ,'THE VARBPT CARDS MUST BE CHANGED IN A SIMILAR MANNER')
 2008 FORMAT(1H1)
C
C PUNCH OUT THE TABLE NAME AND ASSOCIATED VARIABLES
  71  FORMAT(7X, 'THRUPUT ',A8,'(',A8,')')
  72  FORMAT(7X,'THRUPUT ',A8,'(',A8,',',A8,')')
  73  FORMAT(7X,'THRUPUT ',A8,'(',A8,',',A8,',',A8,')')
 971  FORMAT(7X, 'THRUPUT ',A8,'%',A8,'<')
 972  FORMAT(7X,'THRUPUT ',A8,'%',A8,',',A8,'<')
 973  FORMAT(7X,'THRUPUT ',A8,'%',A8,',',A8,',',A8,'<')
 871  FORMAT (4X,'COMMENT, BK PT LIST NO. ',     I3)
 872  FORMAT (4X,'COMMENT, BK PT LIST NOS.',     I3,6X,I3)
 873  FORMAT (4X,'COMMENT, BK PT LIST NOS.',     I3,6X,I3,6X,I3)
      DO 100 N=1,NUMTBL
      NTD=NTDIM(N)
      GO TO(110,120,130),NTD
 110  J=NJ(1,N)
      PRINT 71,TNME(N),VARNM(J)
      WRITE (6,871) NK(1,N)
      WRITE(7,971)TNME(N),VARNM(J)
      GO TO 100
 120  J1=NJ(1,N)
      J2=NJ(2,N)
      PRINT 72,TNME(N),VARNM(J1),VARNM(J2)
      WRITE (6,872) NK(1,N),NK(2,N)
```

```
      WRITE(7,972)TNME(N),VARNM(J1),VARNM(J2)
      GO TO 100
 130  J1=NJ(1,N)
      J2=NJ(2,N)
      J3=NJ(3,N)
      PRINT 73,TNME(N),VARNM(J1),VARNM(J2),VARNM(J3)
      WRITE (6,873) NK(1,N),NK(2,N),NK(3,N)
      WRITE(7,973)TNME(N),VARNM(J1),VARNM(J2),VARNM(J3)
 100  CONTINUE
C
  51  FORMAT(15X,57A1)
 951  FORMAT(16X,57A1)
C PUNCH ALL THE BREAKPOINTLISTS THAT ARE NOT DOUPLICATED
      DO 200 J=1,NOVARS
C PRINT COMMENT NAME AND NUMBER OF THE TABLE FOR
      JT=JTN(J)
      DO 220 K=1,JT
      NX=NLENTH(J,K)
      NT=NTABLE(J,K)
      PRINT 230,K,VARNM(J)
      WRITE(7,930)K,VARNM(J)
 230  FORMAT(1H ,'*THIS IS BK PT LST NO. ',I3,' FOR VARIABLE ',A8)
 930  FORMAT(    '*THIS IS BK PT LST NO. ',I3,' FOR VARIABLE ',A8)
C PUNCH THE BREAKPOINT NAME
      PRINT 270,VARNM(J)
      WRITE(7,970)VARNM(J)
 270  FORMAT(1H ,A7,   'VARBPT')
 970  FORMAT(    A7,   'VARBPT')
      LST=0
C ASSEMBLE THE BREAKPOINTLIST
      DO 250 L=1,NX
      CALL FTOB(BPL(J,K,L),IB,LGTH )
      DO 260 LG=1,LGTH
      LST=LST+1
 260  NOLIST(LST)=IB(LG)
      IF(L.EQ.NX) GO TO 252
      IF(LST.LT.49) GO TO 250
C REMOVE THE LAST COMMA
 252  NOLIST(LST)=M(14)
      PRINT 951,(NOLIST(LG),LG=1,LST)
      WRITE(7,51)(NOLIST(LG),LG=1,LST)
      LST=0
 250  CONTINUE
 251  FORMAT(16X,'/')
      PRINT 251
      WRITE(7,251)
```

```
 220  CONTINUE
 200  CONTINUE
C
C GO TO THE BEGINNING OF THE FILE
      REWIND 2
      LST=0
C  PUNCH A TABLE ARRAY ONE FOR EACH TABLE
      DO 90   N=1,NUMTBL
      PRINT 9,TNME(N)
      WRITE(7,909)TNME(N)
   9  FORMAT(1H ,A7,'POINTS')
 909  FORMAT(    A7,'POINTS')
      LMAX=NL(3,N)
      KMAX=NL(2,N)
      JMAX=NL(1,N)
      IF(NTDIM(N).EQ.1)KMAX=1
      IF(NTDIM(N).EQ.1.OR.NTDIM(N).EQ.2)LMAX=1
      READ(2) (((F(J,K,L),J=1,JMAX),K=1,KMAX),L=1,LMAX)
      DO 10 L=1,LMAX
      IF(NTDIM(N).NE.3) GO TO 20
  25  FORMAT(1H ,'*         ',A8,'=',F8.5)
 925  FORMAT(    '*         ',A8,'#',F8.5)
      JSUB=NJ(3,N)
      KSUB=NK(3,N)
CPUNCH COMMENT CARD
      PRINT 25,VARNM(JSUB),BPL(JSUB,KSUB,L)
      WRITE(7,925)VARNM(JSUB),BPL(JSUB,KSUB,L)
  20  CONTINUE
      DO 30 K=1,KMAX
      DO 40 JB=1,57
  40  NOLIST(JB)=M(14)
C BLANKING THE LINE
      LST=0
      DO 50 J=1,JMAX
C CONVERT A NUMBER TO A SHORT ALPHA ARRAY
      CALL FTOB(F(J,K,L),IB,LGTH)
C APPEND THE WORDS
      DO 60 LG=1,LGTH
      LST=LST+1
  60  NOLIST(LST)=IB(LG)
      IF(J.EQ.JMAX) GO TO 52
      IF(LST.LT.49)GO TO 50
C  REMOVE THE LAST COMMA
  52  NOLIST(LST)=M(14)
      PRINT951,(NOLIST(LG),LG=1,LST)
      WRITE(7,51)(NOLIST(LG),LG=1,LST)
```

```
      LST=0
50    CONTINUE
30    CONTINUE
10    CONTINUE
      PRINT 251
      WRITE(7,251)
90    CONTINUE
      STOP
      END
```

```
      SUBROUTINE FTOB(AF,IO,LGTH)
C THIS SUBROUTINE CONVERTS A FLOATING POINT NUMBER TO A SHORT ALPHA
C ARRAY WITH A MAXIMUM OF FOUR SIGNIFICANT DIGITS FOLLOWED BY A COMMA
C
C A=INPUT FLOATING POINT NUMBER
C IO=ALPAMERIC OUTPUT ARRAY
C LGTH=LENGTH OF THE OUTPUT ARRAY
      DIMENSION  N(17),IO(20)
      DOUBLE PRECISION A,THOUS,P5,TEN
      DATA THOUS,TEN,P5/999.9,10.,0.5/
      DATA N/'1','2','3','4','5','6','7','8','9','0','-',',','.',' ','/'
     1,'(',')'/
C NEXT CARD TO BE USED INSTEAD OF LAST ONE FOR  026 PUNCHING
C    1,'%','<'/
C N=ALPA ARRAY THAT CONTAINS ALL THE SYMBOLS NEEDED
C
C N(1)=1 ETC N(11)=- N(12)=, N(13)=. N(14)=  (BLANK),N(15)=/,N(16)=(,N(17)=)
C BE SURE THAT THE SYMBOLS ARE ENTERED ACCORDING TO OLD STYLE PUNCH
      LGTH=0
      A=AF
      DA=DABS(A)
      IF(DA      .GT. 9999.9)GO TO 55
      IF(DA.LT.0.0001.AND.DA.NE.0.0)GO TO 55
      IF(A) 10,50,20
C SAVING THE MINUS SIGN
 10   IO(1)=N(11)
      LGTH=1
      A=DA
 20   CONTINUE
C SCALING THE NUMBER TO 4 SIGNIFICANT DIGITS
      KS=0
 30   IF(A.GE.THOUS) GO TO 40
      KS=KS-1
      A=TEN*A
      GO TO 30
C CONVERTING TO INTEGER AND ADDING 1 IF REMAINDER IS GREATER THAN 0.5
 40   NA=A +P5
      GO TO 60
 50   IO(1)=N(13)
      IO(2)=N(10)
      IO(3)=N(12)
      LGTH=3

      RETURN
 55   IO(1)=N(15)
      IO(2)=N(15)
```

```
      IO(3)=N(15)
      IO(4)=N(15)
      IO(5)=N(15)
      IO(6)=N(15)
      IO(7)=N(15)
      LGTH=7
      RETURN
C CONVERT NUMBER TO ALPHA ARRAY
 60   J=0
 65   IF(NA-1000)80,70,70
 70   J=J+1
      NA=NA-1000
      GO TO 65
 80   K=0
 85   IF(NA-100) 100, 90,90
 90   K=K+1
      NA=NA-100
      GO TO 85
100   L=0
 105  IF(NA-10) 120,110,110
 110  L=L+1
      NA=NA-10
      GO TO 105
 120  M=0
 125  IF(NA-1) 140,130,130
 130  M=M+1
      NA=NA-1
      GO TO 125
 140  CONTINUE
      ISCALE = KS+8
      IF(K.EQ.0) K=10
      IF(L.EQ.0) L=10
      IF(M.EQ.0) M=10
      GO TO (209,210,220,230,240,250,270,280),ISCALE
C EX 4051.
 280  IO(LGTH+1)=N(J)
      IO(LGTH+2)=N(K)
      IO(LGTH+3)=N(L)
      IO(LGTH+4)=N(M)
      IO(LGTH+5)=N(13)
      LGTH=LGTH+6
      GO TO 300
C EX 405.1
 270  IO(LGTH+1)=N(J)
      IO(LGTH+2)=N(K)
      IO(LGTH+3)=N(L)
```

```
      IO(LGTH+4)=N(13)
      IO(LGTH+5)=N(M)
      LGTH=LGTH+6
      GO TO 300
C EX 40.51
 250  IO(LGTH+1)=N(J)
      IO(LGTH+2)=N(K)
      IO(LGTH+3)=N(13)
      IO(LGTH+4)=N(L)
      IO(LGTH+5)=N(M)
      LGTH=LGTH+6
      GO TO 300
C EX 4.051
 240  IO(LGTH+1)=N(J)
      IO(LGTH+2)=N(13)
      IO(LGTH+3)=N(K  )
      IO(LGTH+4)=N(L  )
      IO(LGTH+5)=N(M  )
      LGTH=LGTH+6
      GO TO 300
C EX .4051
 230  IO(LGTH+1)=N(13)
      IO(LGTH+2)=N(J  )
      IO(LGTH+3)=N(K  )
      IO(LGTH+4)=N(L  )
      IO(LGTH+5)=N(M  )
      LGTH=LGTH+6
      GO TO 300
C  EX .04051
 220  IO(LGTH+1)=N(13)
      IO(LGTH+2)=N(10)
      IO(LGTH+3)=N(J  )
      IO(LGTH+4)=N(K  )
      IO(LGTH+5)=N(L  )
      IO(LGTH+6)=N(M  )
      LGTH=LGTH+7
      GO TO 300
C EX .004051
 210  IO(LGTH+1)=N(13)
      IO(LGTH+2)=N(10)
      IO(LGTH+3)=N(10)
      IO(LGTH+4)=N(J  )
      IO(LGTH+5)=N(K  )
      IO(LGTH+6)=N(L  )
      IO(LGTH+7)=N(M  )
      LGTH=LGTH+8
```

```
      GO TO 300
 209  IO(LGTH+1)=N(13)
      IO(LGTH+2)=N(10)
      IO(LGTH+3)=N(10)
      IO(LGTH+4)=N(10)
      IO(LGTH+5)=N(J)
      IO(LGTH+6)=N(K)
      IO(LGTH+7)=N(L)
      IO(LGTH+8)=N(M)
      LGTH=LGTH+9
 300  CONTINUE
C COUNT THE NUMBER OF SIGNIFICANT DIGITS
      ISIG=4
C CHECK THAT THE O'S ARE TO THE LEFT OF THE DECIMAL POINT
      IF(IO(LGTH-1).EQ.N(13))GO TO 160
      IF(M -10)160,150,160
 150  ISIG=3
      IF(IO(LGTH-2).EQ.N(13))GO TO 160
      IF(L-10)160,170,160
 170  ISIG=2
      IF(IO(LGTH-3).EQ.N(13))GO TO 160
      IF(K-10) 160,180,160
 180  ISIG=1
 160  CONTINUE
      LGTH=LGTH+ISIG-4
C ADDING THE COMMA
      IO(LGTH)=N(12)
      RETURN
      END
```

APPENDIX IV

INTERPOLATION PROGRAM LISTINGS

The following programs perform table look-up and interpolation:

1. SEARCH
2. FOUT1D
3. FOUT2D
4. FOUT3D

```
      SUBROUTINE SEARCH
C THIS SUBROUTINE SEARCHES THE MASTERLISTS ,DETERMINES  THE LOWER (LSUB)
C NEAREST SUBSCRIPT, AND CALCULATES THE PROPORTIONALITY CONSTANTS PFAC
C THAT ARE NEEDED TO INTERPOLATE THE FUNCTIONS.
C DURING THIS OPERATION THE LIMITS OF THE ARGUMENTS ARE CHECKED
C AND RESET WHEN THEY ARE OUT OF LIMITS, AND A MESSAGE IS PRINTED OF THAT
C FACT.
      COMMON /CATLOG/TNME,VARNM,JTN,NOVARS,NMAST,NBPL,BPL,AMAST,NJ,NK,NL
      COMMON /INTPL/ PFAC,LSUB
      COMMON /ARGS/ ARG
C*********************************************************************
C SIZE OF ARRAYS ARE PROGRAM DEPENDENT
C DIMENSION STATEMENTS FROM THE PRE-PROCESSOR OUTPUT
      REAL*8 VARNM(3)
      DIMENSION JTN(3),NBPL(3,7,2),BPL(3,2,4),NMAST(3)
      DIMENSION AMAST(7,3)
      DIMENSION NJ(3,3),NK(3,3),NL(3,3)
      REAL*8 TNME( 3)
C DIMENSION STATEMENTS FOR COMMUNICATION  BETWEEN SEARCH AND FOUT#D
      DIMENSION ARG(3),LSUB(3),PFAC(3,2)
C*********************************************************************
      DO 130 J=1,NOVARS
      JT=JTN(J)
      NMASTL=NMAST(J)
C CHECK IF THE ARGUMENT IS WITHIN THE LIMITS
      IF(ARG(J).LT. AMAST(1,J) .OR. ARG(J) .GE. AMAST(NMASTL,J)) GO TO 1
C ARG IS WITHIN LIMITS HENCE START THE SEARCH OF THE MASTER LIST
      LSU=LSUB(J)
      IF (LSU .EQ. NMASTL) LSU = LSU - 1
 10   IF(ARG(J) .GE. AMAST(LSU+1,J))GO TO 500
 15   IF(ARG(J) .LT. AMAST(LSU,J))GO TO 510
      LSUB(J)=LSU
      GO TO 20
 500  LSU=LSU+1
      GO TO 10
 510  LSU=LSU-1
      GO TO 15
 20   DO 18 K=1,JT
C CALCULATE THE PROPORTIONALITY FACTORS FOR THE K TH BPL OF THE J TH ARGUMENT
C CALCULATE THE FACTOR ONLY IF THE ARGUMENT IS WITHIN THE RANGE
C OF THE BREAK POINT SUBLIST
      L=NBPL(J,LSU,K)
C  IF OUTSIDE LIMIT OF SUBLIST, DO NOT CALCULATE PFAC,  IT WILL BE TAKEN
C  CARE OF ELSEWHERE.
      IF (L) 18,18,17
 17   PFAC(J,K)=(ARG(J)-BPL(J,K,L))/(BPL(J,K,L+1)-BPL(J,K,L))
```

```
18    CONTINUE
      GO TO 130
1     PRINT 110, VARNM(J) ,ARG(J)
110   FORMAT(1H ,A8    ,'=',F10.5,' OUTSIDE THE MASTER LIST LIMITS,RESET
     1TO LIMIT')
C WHEN THE ABOVE OCCURS, WE SET THE ARG BACK TO THE LIMIT
      IF(ARG(J) .LT. AMAST(1,J)) GO TO 120
C RESET ARG TO THE MAX LIMIT
      ARG(J)=AMAST(NMASTL,J)
      LSUB(J)=NMASTL
      DO 111 K=1,JT
111   PFAC(J,K)=0.
      GO TO 130
C SET THE ARGUMENT TO THE LOWEST VALUE
120   ARG(J)=AMAST(1,J)
      LSUB(J)=1
      DO 121 K=1,JT
121   PFAC(J,K)=0.
130   CONTINUE
      RETURN
      END
```

```
      FUNCTION FOUT1D(F,N)
C FOUT1D  INTERPOLATES A 1 DIMENSIONAL TABLE
C
C F= TABLE NAME,
C N = TABLE NUMBER
      COMMON /INTPL/ PFAC,LSUB
      COMMON /CATLOG/TNME,VARNM,JTN,NOVARS,NMAST,NBPL,BPL,AMAST,NJ,NK,NL
      COMMON /ARGS/ ARG
C***********************************************************************
C SIZE OF ARRAYS ARE PROGRAM DEPENDENT
C DIMENSION STATEMENTS FROM THE PRE-PROCESSOR OUTPUT
      REAL*8 VARNM(  3)                                                 D   4
      DIMENSION BPL(  3,  2,  4)                                        D   5
      REAL*8 TNME(  3)                                                  D   8
      DIMENSION JTN(  3)                                                D   9
      DIMENSION NBPL(  3,  7,  2)                                       D  11
      DIMENSION NMAST (  3)                                             D  12
      DIMENSION AMAST(  7,  3)                                          D  13
      DIMENSION NJ( 3,  3)                                              D  14
      DIMENSION NK( 3,  3)                                              D  15
      DIMENSION NL( 3,  3)                                              D  16
C DIMENSION STATEMENTS FOR COMMUNICATION  BETWEEN SEARCH AND FOUT#D
      DIMENSION ARG(3),LSUB(3),PFAC(3,2)
C***********************************************************************
      DIMENSION F(1)
      NJS=NJ(1,N)
      NKS=NK(1,N)
      LS=LSUB(NJS)
      NS=NBPL(NJS,LS,NKS)
C CHECK THAT ALL ARGUMENTS ARE WITHIN TABLE LIMITS
      IF (NS) 10,12,30
 10   NS = 1
      PRINT 11,N,TNME(N),VARNM(NJS),ARG(NJS)
 11   FORMAT(1H ,'TABLE ',I3,1X,A8,'LOOKUP ',A8,'=',G11.4,' OUTS.RANGE.
     1OUTPUT SET TO LOWER SUBLIST LIMIT')
      FOUT1D = F(1)
      RETURN
 12   PRINT 15,N,TNME(N),VARNM(NJS),ARG(NJS)
 15   FORMAT(1H ,'TABLE ',I3,1X,A8,'LOOKUP ',A8,'=',G11.4,' OUTS.RANGE.
     1OUTPUT SET TO UPPER SUBLIST LIMIT')
 14   LS = LS - 1
      NS = NBPL(NJS,LS,NKS)
      IF (NS) 14,14,16
 16   FOUT1D = F(NS+1)
      RETURN
 30   FOUT1D=F(NS)+PFAC(NJS,NKS)*(F(NS+1)  -F(NS))
```

```
RETURN
END
```

```
      FUNCTION FOUT2D(F,N)
C FOUT2D  INTERPOLATES A 2 DIMENSIONAL TABLE
C
C F= TABLE NAME,
C N = TABLE NUMBER
C M1= DIMENSION OF 1ST ARGUMENT
C M2= DIMENSION OF  2ND ARGUMENT
      COMMON /CATLOG/TNME,VARNM,JTN,NOVARS,NMAST,NBPL,BPL,AMAST,NJ,NK,NL
      COMMON /INTPL/ PFAC,LSUB
      COMMON /ARGS/ ARG
C*************************************************************************
C SIZE OF ARRAYS ARE PROGRAM DEPENDENT
C DIMENSION STATEMENTS FROM THE PRE-PROCESSOR OUTPUT
      REAL*8 VARNM(  3)
      DIMENSION BPL(  3,  2,  4)
      REAL*8 TNME(  3)
      DIMENSION JTN(  3)
      DIMENSION NBPL(  3,  7,  2)
      DIMENSION NMAST (  3)
      DIMENSION AMAST(  7,  3)
      DIMENSION NJ( 3,  3)
      DIMENSION NK( 3,  3)
      DIMENSION NL( 3,  3)
C DIMENSION STATEMENTS FOR COMMUNICATION  BETWEEN SEARCH AND FOUT#D
      DIMENSION ARG(3),LSUB(3),PFAC(3,2)
C*************************************************************************
      DIMENSION F(1)
      KKK1=0
      KKK2=0
      NJ1=NJ(1,N)
      NJ2=NJ(2,N)
      NK1=NK(1,N)
      NK2=NK(2,N)
      L1=LSUB(NJ1)
      L2=LSUB(NJ2)
      PF1=PFAC(NJ1,NK1)
      PF2=PFAC(NJ2,NK2)
C CHECK THAT ALL ARGUMENTS ARE WITHIN TABLE LIMITS
   15 N1=NBPL(NJ1,L1,NK1)
      IF(N1) 10,12,30
 10   PRINT 11,N,TNME(N),VARNM(NJ1),ARG(NJ1)
 11   FORMAT(1H ,'TABLE ',I3,1X,A8,'LOOKUP ',A8,'=',G11.4,' OUTS.RANGE.
     1OUTPUT SET TO LOWER SUBLIST LIMIT')
      N1=1
      GO TO 30
   12 IF(KKK1) 13,13,14
```

```
 13 PRINT 16,N,TNME(N),VARNM(NJ1),ARG(NJ1)
 16 FORMAT(1H ,'TABLE ',I3,1X,A8,'LOOKUP ',A8,'=',G11.4,' OUTS.RANGE.
   1OUTPUT SET TO UPPER SUBLIST LIMIT')
    KKK1=1
    PF1=1.
 14 L1=L1-1
    GO TO 15
 30 N2=NBPL(NJ2,L2,NK2)
    IF(N2) 40,55,60
40  PRINT 11,N,TNME(N),VARNM(NJ2),ARG(NJ2)
    N2=1
    GO TO 60
 55 IF(KKK2) 20,20,21
 20 PRINT 16,N,TNME(N),VARNM(NJ2),ARG(NJ2)
    KKK2=1
    PF2=1.
 21 L2=L2-1
    GO TO 30
60  M1=NL(1,N)
    NA=(N2-1)*M1+N1
    NB=NA+M1
    F1=F(NA)+PF1*(F(NA+1)-F(NA))
    F2=F(NB)+PF1*(F(NB+1)-F(NB))
    FOUT2D=F1+PF2*(F2-F1)
    RETURN
C EQUIVALENT 2 DIMENSIONAL PROGRAMMING OF THE INTERPOLATION
C60 F1=F(N1,N2)+PFAC(NJ1,NK1)*(F(N1+1,N2)-F(N1,N2))
C   F2=F(N1,N2+1)+PFAC(NJ1,NK1)*(F(N1+1,N2+1) -F(N1,N2+1))
    END
```

```
      FUNCTION FOUT3D(F,N)
C FOUT3D  INTERPOLATES A 3 DIMENSIONAL TABLE
C
      COMMON /CATLOG/TNME,VARNM,JTN,NOVARS,NMAST,NBPL,BPL,AMAST,NJ,NK,NL
      COMMON /INTPL/ PFAC,LSUB
      COMMON /ARGS/ ARG
C***********************************************************************
C SIZE OF ARRAYS ARE PROGRAM DEPENDENT
C DIMENSION STATEMENTS FROM THE PRE-PROCESSOR OUTPUT
      REAL*8 VARNM(   3)
      DIMENSION JTN(3),NBPL(3,7,2),BPL(3,2,4),NMAST(3)
      DIMENSION AMAST(7,3)
      DIMENSION NJ(3,3),NK(3,3),NL(3,3)
      REAL*8 TNME( 3)
C DIMENSION STATEMENTS FOR COMMUNICATION  BETWEEN SEARCH AND FOUT#D
      DIMENSION ARG(3),LSUB(3),PFAC(3,2)
C***********************************************************************
        DIMENSION F(1)
      KKK1=0
      KKK2=0
      KKK3=0
      NJ1=NJ(1,N)
      NJ2=NJ(2,N)
      NJ3=NJ(3,N)
      NK1=NK(1,N)
      NK2=NK(2,N)
      NK3=NK(3,N)
      L1=LSUB(NJ1)
      L2=LSUB(NJ2)
      L3=LSUB(NJ3)
      PF1=PFAC(NJ1,NK1)
      PF2=PFAC(NJ2,NK2)
      PF3=PFAC(NJ3,NK3)
C CHECK THAT ALL ARGUMENTS ARE WITHIN TABLE LIMITS
   15 N1=NBPL(NJ1,L1,NK1)
      IF(N1) 10,12,30
 10   PRINT 11,N,TNME(N),VARNM(NJ1),ARG(NJ1)
 11   FORMAT(1H ,'TABLE ',I3,1X,A8,'LOOKUP ',A8,'=',G11.4,' OUTS.RANGE.
     1OUTPUT SET TO LOWER SUBLIST LIMIT')
      N1=1
      GO TO 30
   12 IF(KKK1) 13,13,14
   13 PRINT 16,N,TNME(N),VARNM(NJ1),ARG(NJ1)
   16 FORMAT(1H ,'TABLE ',I3,1X,A8,'LOOKUP ',A8,'=',G11.4,' OUTS.RANGE.
     1OUTPUT SET TO UPPER SUBLIST LIMIT')
      KKK1=1
```

```
      PF1=1.
   14 L1=L1-1
      GO TO 15
   30 N2=NBPL(NJ2,L2,NK2)
      IF(N2) 40,55,60
40    PRINT 11,N,TNME(N),VARNM(NJ2),ARG(NJ2)
      N2=1
      GO TO 60
   55 IF(KKK2) 20,20,21
   20 PRINT 16,N,TNME(N),VARNM(NJ2),ARG(NJ2)
      KKK2=1
      PF2=1.
   21 L2=L2-1
      GO TO 30
   60 N3=NBPL(NJ3,L3,NK3)
      IF(N3) 70,75,90
   70 PRINT 11,N,TNME(N),VARNM(NJ3),ARG(NJ3)
      N3=1
      GO TO 90
   75 IF(KKK3) 80,80,81
   80 PRINT 16,N,TNME(N),VARNM(NJ3),ARG(NJ3)
      KKK3=1
      PF3=1.
   81 L3=L3-1
      GO TO 60
90    M1=NL(1,N)
      M2=NL(2,N)
      MS=M2*M1
      NA=(N3-1)*MS+(N2-1)*M1+N1
       NC=NA+M1
      NE=NA+MS
      NG=NE+M1
      F11= F(NA)+PF1*(F(NA+1)-F(NA))
      F12=F(NC)+PF1*(F(NC+1)-F(NC))
      F21=F(NE)+PF1*(F(NE+1)-F(NE))
      F22=F(NG)+PF1*(F(NG+1)-F(NG))
      F1 = F11+PF2           *(F12-F11)
      F2=F21+PF2           *(F22-F21)
      FOUT3D=F1+PF3*(F2-F1)
      RETURN
      END
```

APPENDIX V

USE OF THE PRE-PROCESSOR

General Information

This appendix provides information to allow users to prepare their input cards and to process them with the table processing package described in this working paper. It is written to be a self-contained users guide. Consequently some of the information is similar to that given in the main body.

The table processing package has been compiled and stored in a job library of the authors' TSS/360 library. A PERMIT command has been issued so that anyone can access the package on a read-only basis. Before the user can access the package, he must issue a SHARE command of the form

```
SHARE DSNAME=MY.TABPROC,USERID=FSMDNW1,OWNERDS=LIB.TABPROC
```

where the underscored data set (job library) name is assigned by the user as desired. This SHARE command needs to be issued only once. It may be placed after the LOGON card for the first computer run. The user must include a DDEF card for his job library (e.g., MY.TABPROC) each time he makes a computer run using any part of the shared package.

USE OF THE PRE-PROCESSOR

The pre-processor set of routines is used to read in aerodynamic coefficient tables, process and store them, and generate auxiliary arrays for use in other parts of the table processing package.

The first step in the process is to have the tables punched into IBM cards in the formats shown in figure V-1. This general format is similar to one commonly used when manually tabulating aero data from graphs. The card data fields are 8 columns wide, although such exceptions as table dimensions on the header cards should be noted. The table entries themselves are in floating point format and, for ease of card punching, are left justified. Note that everything is left justified in the fields except the integer values, which must be right justified. Floating point values must have a decimal point.

The table header cards (cards (3), (7), (20) of figure V-1(a) are set up as follows. The first field of 8 characters is a user's comment area. If desired for future reference, table numbers may be put here. For cataloging purposes, the Pre-processor assigns table numbers sequentially as the tables are read in. User's labeling of table numbers should follow this convention. The next fields contain the table name, up to 3 argument names as needed, and the break-point list lengths in the following order:

Field Columns	Contents
9-16	Table Name
17-24	Horizontal break-point list name
25-32	Vertical break-point list name
33-40	Third break-point list name for 3rd tables
41-44	Length of horizontal break-point list
45-48	Length of vertical break-point list
49-52	Length of third break-point list

Names are alphanumeric and are left justified. Lengths are integer and are right justified. Table names should be different from the function names they represent; for instance,CLT instead of CL. Break-point list names may be the same as the related variable names.

card no.	1–8	9–16	17–24	25–32	33–40	41–44	45–48	49–52	
(1)	3 (5)					no. of arguments			key cards
(2)	ALPHA	DA	MACH			argument names			key cards
(3)	1	CM	ALPHA			1	3	1	1 D table
	α	C_m							
(4)	0.	-.2							
(5)	5.	0.6							
(6)	10.	.8							
(7)	2	CLDA	ALPHA	DA	MACH	3	4	2	3 D table
(8)		.5 = M							
(9)	δ_a \ α	0.	5.	10.					
(10)	-10.	.1	.1	.1					
(11)	0.	.3	.2	.1					
(12)	15.	.501	.302	.4					
(13)	20.	.8	.4	.598					
(14)		.9 = M							
(15)	δ_a \ α	0.	5.	10.					
(16)	-10.	.05	.048	.05					
(17)	.0	.146	.1	.098					
(18)	15.	.46	.15	.2					
(19)	20.	.39	.20	.31					
(20)	3	CLMN	DA	ALPHA		3	3	1	2 D table
(21)	α \ δ_a	-9.	10.	18.					
(22)	0.	0.00012	.2001	.41					
(23)	6.	.2	.6	.8					
(24)	10.	.4	1.0	1.2					

columns: 1, 8, 9, 17, 25, 33, 41, 44, 48, 49, 52, 57, 80

FIGURE V-1(a). Card formats with a sample set of input data. Blank cards were inserted for clarity only.

	α_1	α_2	α_3	α_4	α_5	α_6	α_7	α_8	α_9
	α_{10}	α_{11}	α_{12}	α_{13}	.	.	.		
δ_1	$F_{1,1}$	$F_{1,2}$	$F_{1,3}$	$F_{1,4}$	$F_{1,5}$	$F_{1,6}$	$F_{1,7}$	$F_{1,8}$	$F_{1,9}$
	$F_{1,10}$	$F_{1,11}$	$F_{1,12}$	$F_{1,13}$	.	.	.		
δ_2	$F_{2,1}$	$F_{2,2}$	$F_{2,3}$	$F_{2,4}$	$F_{2,5}$	$F_{2,6}$	$F_{2,7}$	$F_{2,8}$	$F_{2,9}$
	$F_{2,10}$	$F_{2,11}$	$F_{2,12}$	$F_{2,13}$	.	.	.		

FIGURE V-1(b). Extended Format.

If the 1st breakpoint list has more than 9 members this format applies.

The table entries are punched left justified into 8 column fields as in figure VI(a). One dimensional tables are punched so that the first field contains the break-point (argument) value and the second field contains the corresponding function value, cards (4), (5), and (6). On 2 dimensional tables, the first card contains the horizontal break-point list starting in the second field, (21). Following cards have a vertical break-point value in the first field and corresponding function values across the card, (22), (23), (24). If a table has more than 9 horizontal break-point values and related function values, then overflow cards are made as in figure VI(b). Three dimensional tables are made up similar to 2 dimensional ones except that each 2 dimensional group is preceded by a card with the corresponding third break-point value in second field, cards (8) and (14) or figure VI(a).

The format in figure VI must be strictly adhered to. Only a few checks are written into the program itself. Errors in column displacement are avoided for the floating point number entries by using a program control card (program 1) for the 029 card punch as shown in figure V2 and by visually checking off-line listings of the data cards. For the table ID cards, one switches to program 2 of the program card. Program 2 is punched in such a manner that it is correct for 2 digit integers and for the alpha-numeric information for a three dimensional table. Hence, one must insert a leading blank or 0 for one digit integers and skip once or twice on tables with smaller dimensions. As a further restriction the horizontal break-point

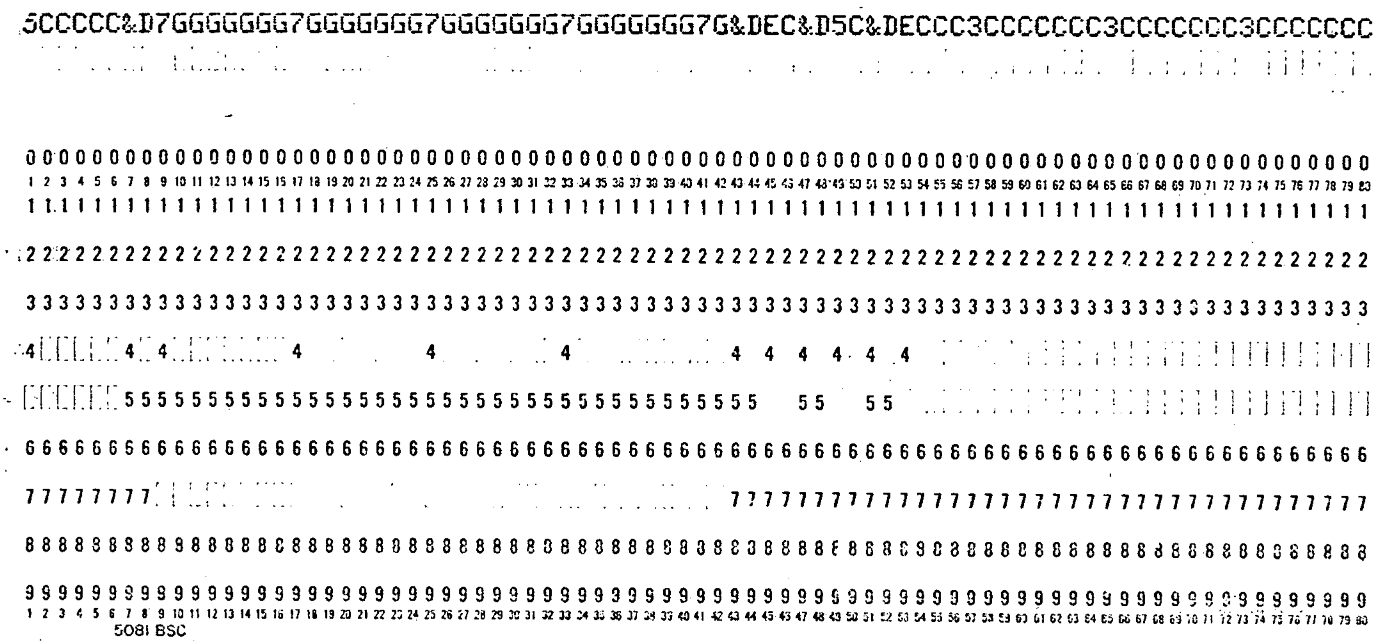


FIGURE V-2.029 Program Control Card

Figure V-1(a)

lists must increase from left to right, (9), (15), (21); the vertical break-point lists must increase downward; and the break-point list elements for the third variable of a three dimensional table must increase in the order of appearance (8), (14).

After the data cards are punched, one or more computer runs are made for initial checking of the tables. Figure V-3 shows a sample deck of TSS/360 control cards for the computer run.

```
(1)    LOGON USERID,JOB,99          JOHN DOE    STOP 99    PH 3999
(2)   DDEF DDNAME=TABLIB,DSORG=VP,DSNAME=MY.TABPROC,OPTION=JOBLIB
(3)   LOAD TREAD$$
(4)   DDEF DDNAME=FT02F001,DSORG=VS,DSNAME=TABLES
(5)   DDEF DDNAME=FT07F001,DSORG=VS,DSNAME=DIMREAD
(6)   CALL TREAD$$
       &INPUT
(7)          IPUNCH=1,
             IPUNCH=0,
       &END
      ********************
      *          **
(8)   *     TABLE DATA CARDS GO HERE
      *          **
      ********************
(9)   %END
(10)   LOGOFF
```

Figure V-3. TSS/360 control cards for computer run using the Pre-processor

Unerlined words should be changed to conform to the user's naming conventions. Card (1) is the standard users LOGON, card. Card (2) defines the user's job library which contains the table processing package. The data set name (DSNAME) should be identical to that specified by the user in the SHARE command. The data set name in

card (4) specifies where the processed tables and related data are to be stored. Card (5) names the data set where the DIMENSION and READ card images are to be stored. The second of the NAMELIST "INPUT" cards (7) determines whether the tables and the DIMENSION and READ card images are to be stored (IPUNCH=1) or not stored (IPUNCH=0). The user's table data cards go at (8). The %END at card (9) must be present to signal the end of the data cards.

It is suggested that off-line listings of the data cards be examined for such things as blank cards, duplicate or missing cards, and the use of wrong columns. Further, it is suggested that the user make computer runs with IPUNCH=0 to conserve computer time until all of his tables and the auxiliary arrays are printed out correctly. Some errors which may cause premature run termination include missing cards in a table and incorrect dimensions on a table header card. After all tables have been printed out, they should be examined carefully for errors. Error message printouts should be noted and their causes remedied. Causes of errors can be determined by examination of the printout and the data cards.

After all apparent errors are corrected, another computer run should be made with IPUNCH=1 to store the tables and the DIMENSION and READ card images in their respective data sets. The DIMENSION and READ cards can be punched using the TSS command

PUNCH DSNAME=DIMREAD,STARTNO=1,ENDNO=80

where the data set name is the same as specified on the DDEF card (5)

of figure V-3. A printed listing of the DIMENSION and READ statements may be obtained by using the TSS command

PRINT DSNAME = DIMREAD, PRTSP = 1,

STATION = RMT02,

where RMT02 is for RJE station 2. These commands can be put before the LOGOFF card (10), run as a separate batch job, or entered conversationally from a terminal. The DIMENSION and READ cards should have interpreted and separated according to type for use in the plotting main program. They are numbered to aid in keeping the READ cards in order.

APPENDIX VI

USE OF THE PLOTTING PROGRAM

The plotting program is written to provide hardcopy plots of 1,2, or 3 dimensional tables on the S-C 4020 plotter. All necessary subroutines are shared in the TSS job library described in Appendix V. The main program, however, is data dependent and must be changed by the user to include DIMENSION and READ cards for his stored tables. A copy of the main program may be obtained from the anthors. The required DIMENSION and READ cards are obtained during use of the pre-processor as described in Appendix V. Comment cards in the main program indicate where these cards are to be placed. The main program may be compiled separately or during the plot job.

The plotting program provides the user with several options for plotting the tables. The options and data cards to specify them are described with reference to figure VI-1. Card (1) transmits information to identify the plot job and to specify the user's tape number. Columns 1 to 68 are printed on the first and last frame of the plot job. Columns 69 to 72 are printed on the upper right hand corner of each plot frame. The user must have a 7 track magnetic tape assigned to him for this purpose. The plot option cards, (2) through (6), specify the table to be plotted, whether the variable order for 2 or 3 dimensional tables to be plotted differs from the order in storage and, if so, what order is desired. The other option specifies special scaling limits desired. Special plot instructions can be largely eliminated by following certain corrections for the table construction as explained in the section

Card No.	User's name, mail stop, etc. for plot identification	Job ID	Plot tape Reel No.
	1 — 68	69 — 72	73 — 80
(1)	JOHN DOE STOP 99 PH 3999 PLOT JOB T999	X99	99999

(a) Plot identification card (must be first data card).

Card No.	Plot option IORDER	Table No.	Variable order option Var. 1	Var 2	Var. 3	Scaling Options: Function scaling min.	Function scaling max.	Var. 1 scaling min.	Var. 1 scaling max.
	1 — 8	9 — 16	17	25	33	41	49	57	65
(2)	0	1							
(3)	0	2							
(4)	1	2	DA	MACH	ALPHA	0.	1.		
(5)	0	3				0.	2.	-10.	20.
(6)	-1								

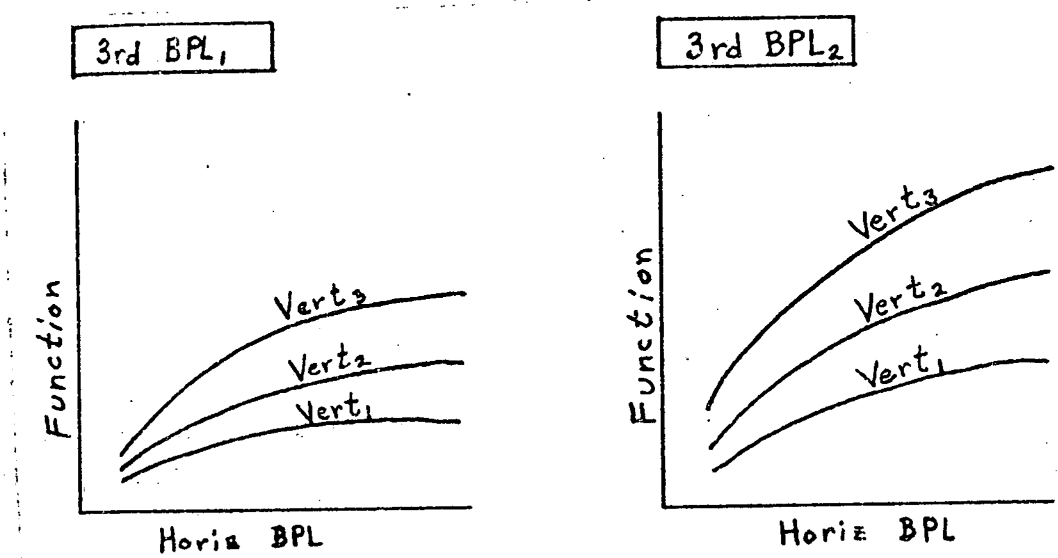
(b) Table plot option cards.

FIGURE VI-1. EXAMPLE OF USER SUPPLIED INFORMATION FOR TABLE PLOTTING

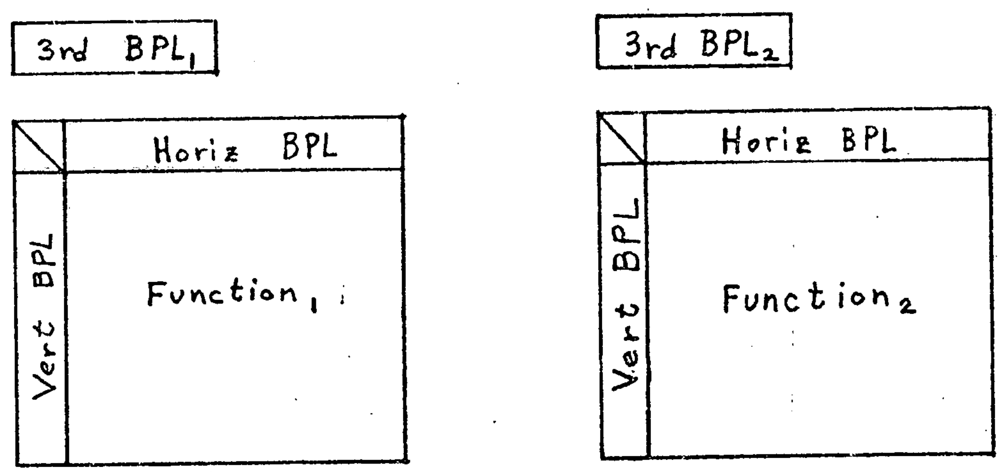
"Considerations for the Preparation of Tables".

A logical variable called IORDER can be set to either -1, 0, or +1 (column 8). If IORDER is equal to -1, all of the tables are plotted in the order in which they have been stored on the disc by the table pre-processor, see figure VI-2. In this case variable 1 is on the abscissa, variable 2 is always plotted as the fixed parameter for a given curve, and variable 3 is always plotted as fixed parameter for each set of curves for 3-dimensional tables. The scaling is automatic. If only a few tables are to be plotted from all the tables that are stored, IORDER may be set equal to 0. In this case more specifications must be given. The second number (right justified column 16) will be the table number. As an optional feature, the next two numbers define the maximum and/or minimum value of the function to be plotted on the ordinate (columns 41 and 49) and the last two values designate the minimum and maximum of the first argument that is plotted as the abscissa (columns 57 and 65). Omission of these values results in automatic scaling of the data points as stored. No scaling, of course, is needed on the second and third arguments. As mentioned before, the arguments may not have been stored in the order as one wants to plot them out. In this case the logical variable IORDER is set equal to 1. In addition to the information that is needed when IORDER equals 0, one also must read in the order of the variables to be plotted by giving the names of the 2 or 3 variables in the proper order (columns 17,25, and 33 as on card (4)).

2



(a) Graphical data



(b) Tabulated data

FIGURE VI-2. RELATIONSHIPS BETWEEN BREAKPOINT LISTS AND STANDARD PLOTTING CONVENTION

The TSS control cards required for the plotting computer run are shown in figure VI-3.

```
(1)   LOGON USERID,JOB,99(7)    JOHN DOE   STOP 99    PH 3999
    ERASE SOURCE.MYPLOT$$ ;   FTN MYPLOT$$,ISD=N
    ******************************
    *           **
(2) *    SUPPLIED MAIN PROGRAM WITH USER'S DIMENSION AND READ CARDS INSERTED
    *    GOES HERE FOR COMPILATION
    *           **
    ******************************
(3) DDEF DDNAME=TABLIB,DSORG=VP,DSNAME=MY.TABPROC,OPTION=JOBLIB
(4) AMES SC4020
(5) LOAD MYPLOT$$
(6) DDEF DDNAME=FT02F001,DSORG=VS,DSNAME=TABLES
(7) MTMSG
    PLEASE OBTAIN 99999 7-TRACK WITH RING
(8) CALL MYPLOT$$
    ******************************
    *           **
(9) *    PLOT OPTION CARDS GO HERE
    *           **
    ******************************
(10) %END
(11)  LOGOFF
```

Figure VI-3. Control Cards for Plotting Job.

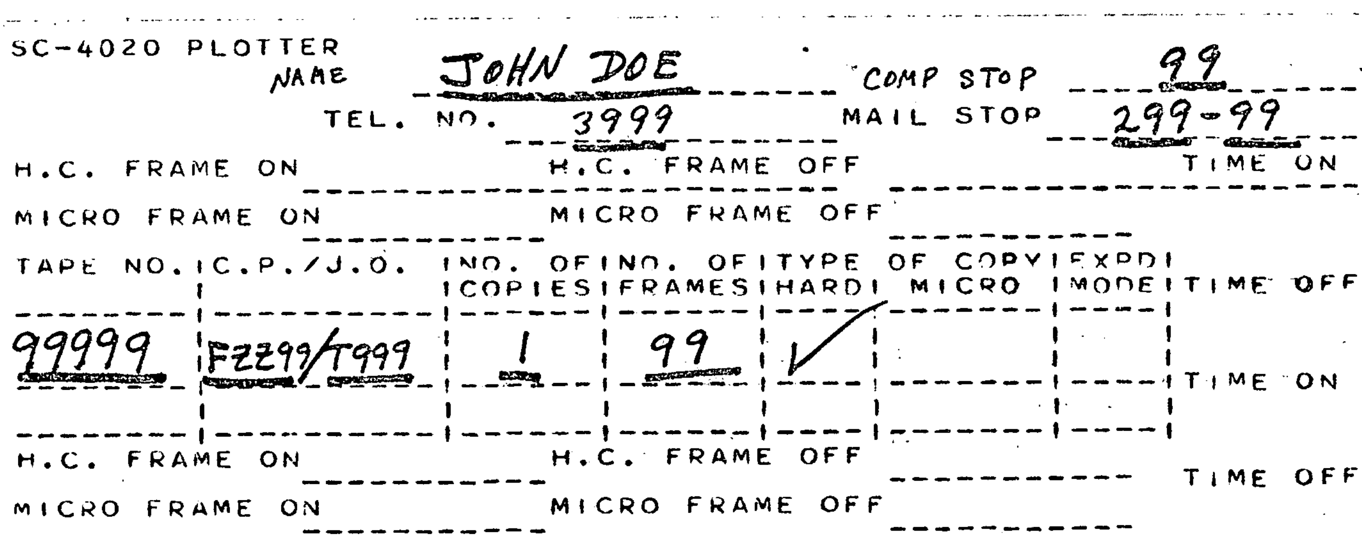
Underlined parts of the cards are to be changed by the user as necessary. An estimate of time required, for the LOGON card, can make based on the authors' experience in plotting the number of tables below during 1 job run.

No. of Tables	Time (sec)
3	15
32	132

Use of the plotting options affects the time used, so it is suggested that 10 - 30% additional time be specified to avoid premature termination of the job. Portions of a job cannot be saved if the tape writing job is terminated due to underestimation of time. The plotting main program, with the user's DIMENSION and READ cards from the pre-processor, are put at (2) with the FTN card for compilation. The DDEF card (3) specifies the user's job library which contains the shared program modules. Card (4) provides access to the Ames library of plotting routines. The LOAD card (5)

must specify the main program named at (2). The user's stored tables to be plotted are specified by card (6). Card (7) specifies the user's 7-track magnetic tape. Card (8) starts writing the plotting tape. The plot identification card and plot option cards described previously go at (9). Card (10) signals the end of the data cards. The printer listing received after the job run will specify which tables were plotted and the number of plot frames made.

The plot run described above generates plotting data on the user's magnetic tape. To get hardcopy plots made, the user must send a card, as in figure VI-4, to the operations room, N233. The card is self-explanatory. Off-line production of the plots from magnetic tape takes approximately 24 hours.



SC-4020 PLOTTER

NAME JOHN DOE COMP STOP 99

TEL. NO. 3999 MAIL STOP 299-99

H.C. FRAME ON H.C. FRAME OFF TIME ON

MICRO FRAME ON MICRO FRAME OFF

TAPE NO.	C.P./J.O.	NO. OF COPIES	NO. OF FRAMES	TYPE OF COPY HARD	MICRO	EXPD MODE	
							TIME OFF
99999	FZZ99/T999	1	99	✓			TIME ON

H.C. FRAME ON H.C. FRAME OFF TIME OFF

MICRO FRAME ON MICRO FRAME OFF

Figure VI-4. Request for SC 4020 Plots.

APPENDIX VII

USE OF THE EAI 8400 DATA CONVERSION PROGRAM

This appendix describes the use of the programs which convert the tables stored on disc by the Pre-processor to a punched card format which can be read directly by the EAI 8400 computer. Only a small amount of manual work, as described, is necessary on the punched cards.

The main program is data dependent and must be recompiled by the user. A card deck copy of it may be obtained from the authors. A set of the DIMENSION and READ cards produced by the Pre-processor is inserted in the main program at the place denoted by comment cards. The main program may then be compiled alone or as part of the computer run to punch "8400" cards.

```
(1)   LOGON USERID,JOB,99        JOHN DOE    STOP 99    PH 3999
(2)  ERASE SOURCE.MYP84$$;    FTN MYP84$S,ISD=N
     ********************************
     *          **
(3)  *    SUPPLIED MAIN PROGRAM WITH USER'S DIMENSION AND READ CARDS INSERTED
     *    GOES HERE FOR COMPILATION
     *          **
     ********************************
(4)  DDEF DDNAME=TABLIB,DSORG=VP,DSNAME=MY.TABPROC,OPTION=JOBLIB
(5)  LOAD MYP84$$
(6)  DDEF DDNAME=FT02F001,DSORG=VS,DSNAME=TABLES
(7)  DDEF DDNAME=FT07F001,DSORG=VS,DSNAME=CRD84
(8)  CALL MYP84$$
(9)  PUNCH DSNAME=CRD84,STARTNO=1,ENDNO=80
(10)  LOGOFF
```

Figure VII-1. Control cards for punching data cards for EAI 8400.

The TSS control cards used compile the main program and to execute it are shown in Figure VII-1. Underlined parts are to be changed by the user as necessary. Card (1) is the user's LOGON card. Card (2) is for compilation of the main program which is placed at (3). Its module name may be chosen by the user and used on cards (2), (5), and (8). The DDEF card (4) specifies the user's job library which contains the shared program modules. Card (6) specifies the user's tables which were stored by the Pre-processor. Card (7) specifies the dataset where card images of the converted tables are to be stored. These cards are punched by card (9), which may be used here, run separately, or entered from the conversational terminal.

The printed output from the computer run gives card image printouts which should be examined by the user. All breakpoint and table values punched out will be between 0.0001 and 9999.9. If a table entry should be outside this range, the number will be replaced by a series of slashes (//////), which can be easily recognized and can be replaced by the proper number. Further information about changing the punched cards to obtain a running deck for the 8400 program is given in the main body of the report.

APPENDIX VIII

THE APPLICATION OF THE TABLE LOOKUP PROGRAMS

The use of the table lookup programs is illustrated by means of an example. The tables to be interpolated are those saved and plotted previously. The sample program is shown in figure VIII-1. First, the DIMENSION and READ cards from the table pre-processor must be inserted in the simulation program to make the tables and auxiliary arrays available to the simulation program (cards 500 to 3800, not shown on the figure). In addition, the common blocks "ARGS" and "CATLOG" must be put into the user's simulation program. Before calling the SEARCH routine, the table arguments must be placed in the proper order into the array ARG (cards 4400 to 4600). SEARCH resets the value of any member of the array ARG to the closest tabulated limit if the argument was outside the tabulated range of the master list. A statement is then printed to this effect. Figure VIII-2 lines 9 and 10 are examples of this output. The lower limit for APLHA in the tables was 0. and the upper limit for MACH was 0.9. Note that SEARCH does not reset the actueal arguments ALPHA, MACH and DA as shown in line 12 of figure VIII-2. Other computations in the simulation program would therefore be based on the actual values of the arguments. This non-reset of actual arguments feature of the simulation can be traded for a small amount of increased computational efficiency by putting the actual argument list in the labelled common block ARGS in the proper order: COMMON/ARGS/ALPHA, DA, MACH

```
      REAL MACH
      COMMON /ARGS/ ARG(3)
      COMMON /CATLOG/TNME,VARNM,JTD,NOVARS,NMAST,NBPL,BPL,AMAST,[illegible]
C  DIMENSION AND READ CARDS FROM PRE-PROCESSOR PROGRAM

C  SAMPLE INPUT
  100 READ (5,1,END=500) ALPHA,DA,MACH
    1 FORMAT (3F6.3)
C  PUT ARGUMENTS INTO COMMON ARRAY IN THE SAME ORDER AS ENTERED INTO
C     THE DATA STORAGE PROGRAM
      ARG(1) = ALPHA
      ARG(2) = DA
      ARG(3) = MACH
C  FIND BREAK-POINT LIST SUBSCRIPTS AND COMPUTE PROPORTIONALITY FACTORS
      CALL SEARCH
C  INTERPOLATE THE TABLES
C     THE FUNCTION ARGUMENTS ARE TABLE NAME AND NUMBER
      CMA = FOUT1D(CM,1)
      CLD = FOUT3D(CLDA,2)
      COR = FOUT2D(CLMN,3)
C  PRINT OUT THE RESULTS
      WRITE (6,2) ALPHA,DA,MACH,CMA,CLD,COR
    2 FORMAT ('0ALPHA = ',F6.3,'     DA = ',F6.3,'     MACH = ',F6.3/
     1  '0CMA = ',F6.3,/'0CLD = ',F6.3,/'0COR = ',F6.3)
      GO TO 100
  500 STOP
      END
```

Figure VIII-1. Sample interpolation program.

```
ALPHA =   5.100     DA =   0.100     MACH =   0.510

CMA =  0.604

CLD =  0.196

COR =  0.347

ALPHA =   2.000     DA =   4.000     MACH =   0.700

CMA =  0.120

CLD =  0.243

COR =  0.249
ALPHA     =  -1.00000 OUTSIDE THE MASTER LIST LIMITS,RESET TO LIMIT
MACH      =   1.00000 OUTSIDE THE MASTER LIST LIMITS,RESET TO LIMIT

TABLE    3 CLMN     LOOKUP DA            = -9.500     OUTS.RANGE.  OUTPUT SET TO
 LOWER SUBLIST LIMIT
ALPHA = -1.000     DA = -9.500     MACH =   1.000

CMA = -0.200

CLD =  0.055

COR =  0.000
```

Reproduced from best available copy.

Figure VIII-2. Outputs from the sample interpolation program.

The function values are obtained when needed by function calls as shown on cards 5100 to 5300 in figure VIII-1. Lines 1 to 8 on the next figure show the argument values and results from the table lookup when all arguments are within the proper limits. If an argument is outside the proper limits of a specific table and within the range of the master list, a call to an FOUTxD function will result in an output as shown on line 11, figure VIII-2 and the function value will be set to the proper sublist limit as indicated on line 14.

The structure of the NBPL list makes the above check very efficient. The appropriate NBPL element is tested with an algebraic IF statement. When the NBPL element, which normally determines the subscript for the breakpoint list, is negative, the argument is below the range of the breakpoint list for the given table. When the NBPL is '0' the argument is above the range of the breakpoint list. In either case the following information is printed out: the number and name of the table and the name and the value of the argument which exceeds the limits of the breakpoint list.

In some cases, it may be desirable to suppress the above error messages. To do this one can add a two value dummy table, which will not be used in the actual program, but which serves to establish a breakpoint list for the parameter in question that spans the whole range of the parameter. An example might be the parameter 'altitude' for ground effect calculations. A second method to remedy the situation which is not quite as efficient, is to use logic statements in the running

program to permit ARC(J) for the given parameter to take on values only within the specified limits.

The table lookup routines are data dependent and must therefore be recompiled by the user after insertion of the proper DIMENSION statements. Card deck copies of these subprograms-SEARCH, FOUTID, FOUT2D, FOUT3D-may be obtained from the authors.

All 4 of these subprograms have identical COMMON blocks and DIMENSION statements, as shown in figure VIII-3. Most of the DIMENSION statements are provided by the pre-processor as described in appendix V. The sizes of the arrays ARG, LSUB, and PFAC are dependent on the sizes of VARNM and BPL as shown by the arrows in figure VIII-3. The values of the array LSUB must be initialized to 1 at load-time. This may be done in a BLOCK DATA subprogram as in figure VIII-4 or by a DO loop in the user's initialization program.

```
      COMMON /INTPL/ PFAC,LSUB
      COMMON /CATLOG/TNME,VARNM,JTN,NOVARS,NMAST,NBPL,BPL,AMAST,NJ,NK,NL
      COMMON /ARGS/ ARG
C*********************************************************************
C SIZE OF ARRAYS ARE PROGRAM DEPENDENT
C DIMENSION STATEMENTS FROM THE PRE-PROCESSOR OUTPUT
      REAL*8 VARNM(  3)
      DIMENSION BPL(  3,  2,  4)
      REAL*8 TNME(  3)
      DIMENSION JTN(  3)
      DIMENSION NBPL(  3,  7,  2)
      DIMENSION NMAST (  3)
      DIMENSION AMAST(  7,  3)
      DIMENSION NJ(  3,  3)
      DIMENSION NK(  3,  3)
      DIMENSION NL(  3,  3)
C DIMENSION STATEMENTS FOR COMMUNICATION  BETWEEN SEARCH AND FOUT#D
      DIMENSION ARG(3),LSUB(3),PFAC(3,2)
C*********************************************************************
```

Figure VIII-3. COMMON and DIMENSION statements used in table lookup subprograms SEARCH, FOUT1D, FOUT2D, FOUT3D.

Reproduced from best available copy.

```
      BLOCK DATA
      COMMON /INTPL/ PFAC,LSUB
      DIMENSION LSUB(3),PFAC(3,2)
      DATA LSUB/3*1/
      END
```

Figure VIII-4. Initialization of LSUB array.